Handleiding Möbius Randomiseren

Handleiding Möbius Randomiseren

Contents

List of Figures

1 Randomiseren met Möbius

Update sept. 2021

1.1 Inleiding

Waarom zou u uw digitale vraagstukken gaan randomiseren?

- Vraagstukken die gerandomiseerd zijn, kunnen vaker gebruikt worden!
  Immers als de student exact dezelfde vraag krijgt als eerder, dan herkent hij deze en weet bij voorbaat het antwoord al. Vaak is de herkenning van een plaatje al genoeg om te weten wat er gevraagd wordt en dan is het antwoord vaak uit het geheugen op te roepen. De vraag wordt dan nauwelijks meer als geheel gelezen.
- Om fraude te voorkomen!
  Elke student krijgt steeds een iets andere vraag aangeboden, zodat het overnemen van antwoorden tot een minimum beperkt wordt.
- Als u een hele familie vragen in één vraag verenigt, kan uw database met vragen drastisch beperkt worden en maakt het geheel overzichtelijker.
- Het bespaart een hoop werk!

Vraagstukken die gerandomiseerd zijn, kunnen vaker gebruikt worden!
Het helpt al als er bij een bepaald plaatje steeds verschillende vragen gesteld worden. Het kan soms zelfs wel eens dezelfde vraag zijn die iets anders geformuleerd is, zodat het de student dwingt om opnieuw te lezen en na te denken. Ook is het handig om te beschikken over meer plaatjes die iets verschillend zijn en waarover wellicht dezelfde vraag gesteld kan worden. Het plaatje fungeert dan niet als trigger van het geheugen.
Het mooist is natuurlijk om plaatjes én tekst én getallen én eventueel formules allemaal steeds iets te veranderen. Denkt u ook eens aan het omwisselen van de gegevens van vraag en antwoord.
Verschillende studenten krijgen steeds verschillende vragen en de vragen zijn vaker te gebruiken en uw *Question Repository* wordt aanmerkelijk minder omvangrijk!
Zelfs bij het vraagtype *Multiple Choice* is het verstandig om uw vragenbank eens te herzien en de vragen proberen te randomiseren wat betreft de tekst, de illustraties en al dies meer. Maak bijvoorbeeld een drietal verschillende formuleringen van de tekst van uw vraag waarbij elke formulering in feite hetzelfde is. Het geeft ieder keer een andere uitstraling aan uw vraag.
Nog mooier is het om meer goede antwoorden en meer foute antwoorden in het *Algorithm* voor te bereiden en daar steeds een greep uit te doen.
In de volgende paragrafen worden de verschillende aspecten van randomisering toegelicht.

In paragraaf *Rondomisering met switch (page 2)* gaat het veelal over tekst en plaatjes.

In paragraaf *Randomisering met berekeningen (page 29)* gaat het veelal over berekeningen, getallen, formules en dynamische plaatjes. Iets voor bèta's dus.

In Paragraaf *Overzicht Permutaties (page 47)* staat een overzicht van de codes die gebruikt worden voornamelijk bij de functie switch.

In paragraaf *Overzicht Randomvariabelen (page 52)* vindt u allerlei mogelijkheden om niet alleen tekstuele randomisatie te maken maar ook functies voor randomisatie van getallen die in het *Algorithm* kunnen worden voorbereid. In die paragraaf vindt u ook informatie over de *Designer* van het *Algorithm*.

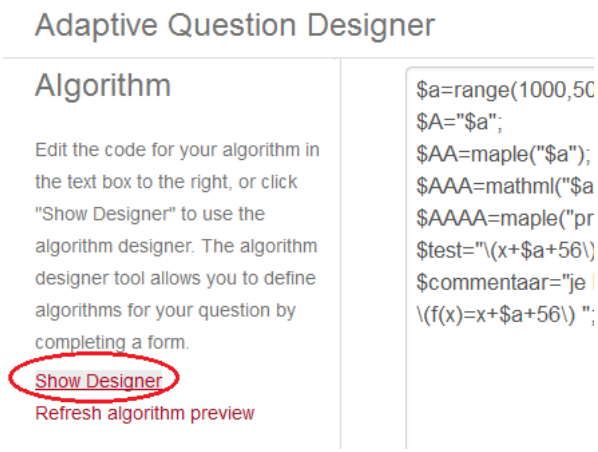


Figure 1.1: Voor het maken van variabelen kan de Designer van het Algorithm gebruikt worden

1.2 Randomisering met switch

Bij tekstuele randomisering is de meest voorkomende functie in het *Algorithm* de functie `switch`.
Met de volgende definitie van kleur wordt er bijvoorbeeld steeds gewisseld tussen de drie kleuren rood, wit en blauw. Deze variabele $kleur kan dan overal in het hele vraagstuk gebruikt worden.

```
$kleur=switch(rint(3),"rood","wit","blauw");
```

Bij deze functie `switch` begint de telling altijd bij 0. Dat wil zeggen dat de eerste keuze in dit geval de kleur rood is als de index gelijk is aan 0.
De functie `rint(3)` wil zeggen: 0 of 1 of 2 (tus tot aan 3 en het getal 3 niet meer meegerekend). In feite dus drie mogelijkheden.
De woorden rood, wit en blauw staan tussen dubbele aanhalingstekens omdat het tekst is. Als het getallen zouden zijn, dan kunnen de aanhalingstekens weggelaten worden, maar dan maakt het nog wel even uit in welke gedaante u de getallen wilt hebben. Dus hou de aanhalingstekens in de gaten. Zie ook paragraaf *(page 31)*.
De functie `switch` is in veel gevallen krachtig genoeg om heel veel mee te kunnen bereiken. Met een vooraf gedefinieerde index kunt u de verschillende variabelen die met elkaar verband houden, in de pas laten lopen. Eventueel kan deze functie ook nog genest worden, zoals in het voorbeeld van *Figure 1.19 (page 12)*.
Niet alleen tekst maar ook getallen of plaatjes kunnen door middel van de functie `switch` door elkaar gehusseld worden.
Voorbeelden van randomiseren van plaatjes vindt u in paragraaf *Vraagtype Matching met plaatjes* *(page 18)* en paragraaf *Drop-down menu's bij gerandomiseerde plaatjes* *(page 24)*.

1.2.1 Blanks (vraagtype List)

1.2.1.1 Teksten wisselen

Bij het volgende voorbeeld wisselt steeds de tekst in de vraag, terwijl het antwoord steeds hetzelfde is.

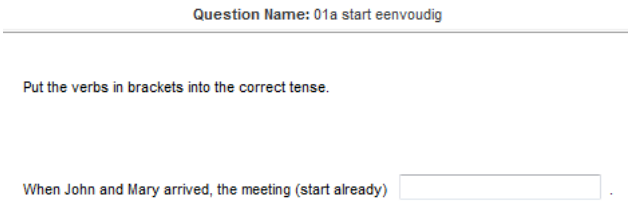


Figure 1.2: Wisselen van tekst in de vraag

Het bijbehorende *Algorithm* is:

```
$meeting=switch(rint(8),"meeting","congress","lecture","explanation","concert","introduction", "performance","presentation");
$John=switch(rint(10),"John","Mary","Jasmine","my brother","David","Sylvia","they","my colleague","John and Mary","we");
```

In de vraag hoeft dan alleen de variabele aangeroepen te worden: When $John arrived, the $meeting (start already).

Het antwoord is in dit geval steeds hetzelfde en kan met het vraagtype *List* worden gecheckt met *Exact text match*.

1.2.1.2 Teksten en antwoorden in de pas

In het volgende voorbeeld wordt de tekst van de vraag steeds aangepast en de antwoorden zijn dan ook steeds verschillend en lopen met elkaar in de pas. Een en ander wordt helemaal voorbereid in het *Algoritm.*

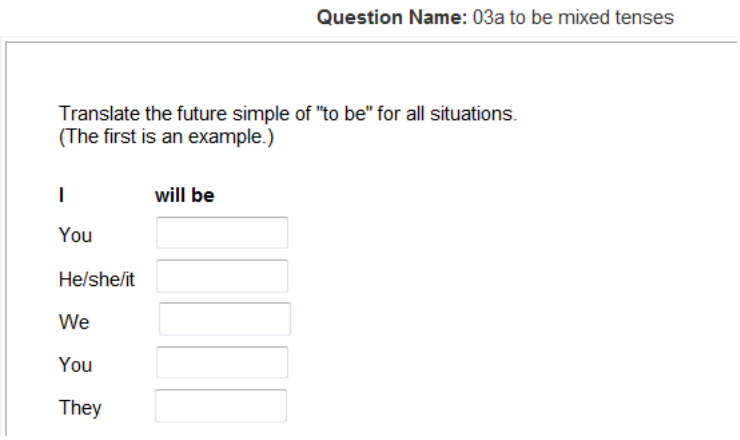


Figure 1.3: Tekst en antwoord in de pas

Het bijbehorende *Algoritm* is:

```
$index=rint(5);
$tense=switch($index,"present tense (simple present)","past tense (simple past)","present perfect tense","past perfect tense","future simple");
$I=switch($index,"am","was","have been","had been","will be");
$you=switch($index,"are","were","have been","had been","will be");
$He=switch($index,"is","was","has been","had been","will be");
$We=switch($index,"are","were","have been","had been","will be");
$You=switch($index,"are","were","have been","had been","will be");
$They=switch($index,"are","were","have been","had been","will be");
```

De vervoegingen van het werkwoord "to go" loopt voor elke persoonsvorm steeds in de pas met de tijdsaanduiding $tense. Dat in de pas lopen heeft te maken met de van te voren vastgestelde $index die voor alle situaties dan steeds dezelfde zal zijn.
Er is slechts één antwoord goed en dat moet in een open tekstveld worden ingevuld.
De vraag is steeds: Translate the $tense of "to be" for all situations.
De antwoordvelden zijn met het vraagtype *List* aangemaakt met *Exact text match.*

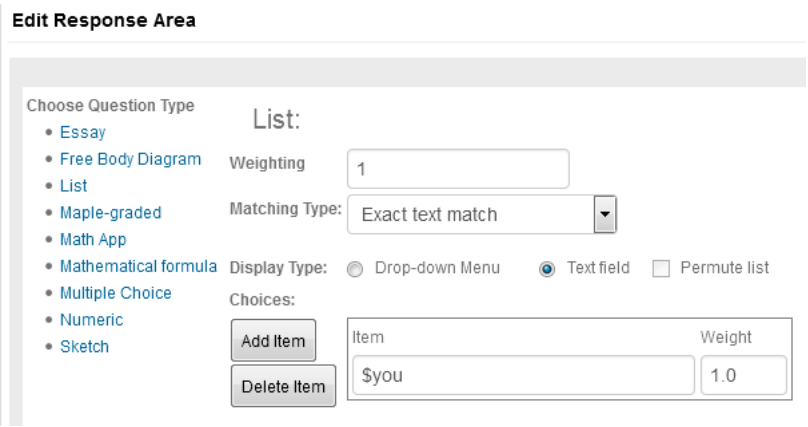


Figure 1.4: Antwoord voorbereid in het Algorithm

1.2.1.3 Meer antwoorden goed

Bij het volgende voorbeeld is er een open tekstveld waar de student de vervoeging van het Engelse werkwoord "to work" moet invoeren, passend in de steeds wisselende context. Het bijbehorende antwoord is ook steeds wisselend.

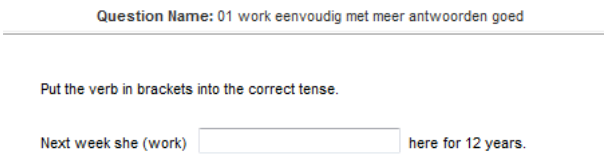


Figure 1.5: Open tekstveld met vraagtype List en meer antwoorden goed

In het bijbehorende *Algorithm* wordt ingevoerd:

```
$index=rint(2);
$time=switch($index,"On monday last week","Next week");
$ww1=switch($index,"had worked","will have worked");
$ww2=switch($index,"had worked","will have been working");
```

Er worden twee mogelijke antwoorden $ww1 en $ww2 voorbereid. In het geval "On monday last week" is er maar één goed antwoord (dus twee dezelfde) en in het geval "Next week" zijn er twee verschillende mogelijke antwoorden.
Met het vooraf definiëren van de variabele $index lopen de drie variabelen $time, $ww1 en $ww2 alle drie met elkaar in de pas.

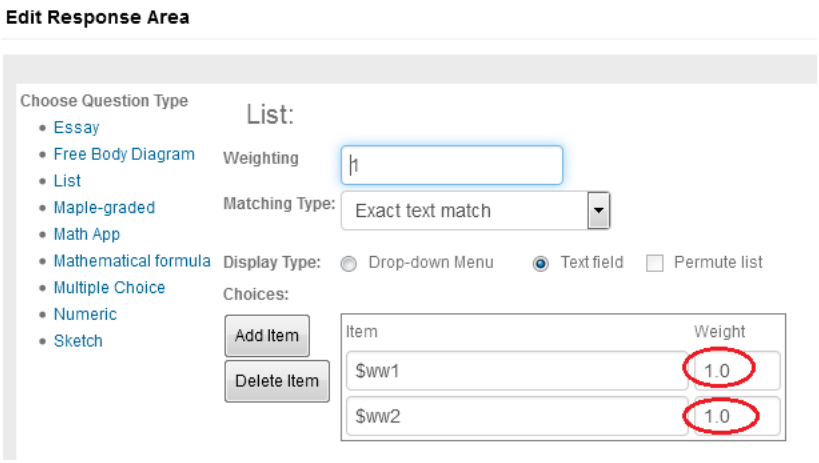


Figure 1.6: Twee antwoorden goed bij vraagtype List

In bovenstaande figuur is te zien dat het vraagtype *List* is gekozen waarbij beide antwoorden goed gerekend worden (Weight = 1.0) en dat er *Text field* is ingesteld, dus een open invulveld (blank) met *Exact text match*.
In de feed back wordt altijd het eerste goede antwoord gegeven als de student het fout heeft ingevuld. Dus in geval van meer goede antwoorden is het raadzaam even goed te bekijken welk goede antwoord u bovenaan wilt hebben staan.

Het wisselen van tekst kan zeer uitgebreid zoals het volgende voorbeeld laat zien:

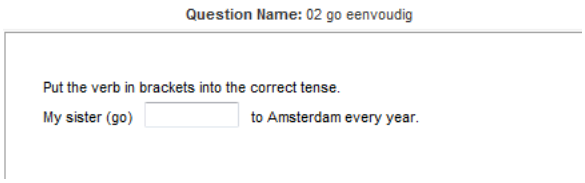


Figure 1.7: Wisselen van tekst en bijbehorend antwoord, mogelijk meer antwoorden goed

Het bijbehorende *Algorithm* is wat ingewikkelder.

```
$habit=switch(rint(4),"every week","every month","every year","every summer");
$past=switch(rint(3),"yesterday","last month","last year");
$future=switch(rint(2),"next week","next month");
$index1=rint(3);
$tijdsbepaling=switch($index1,"$habit","$past","$future");
$index2=rint(3);
$you=switch($index2,"You","My sister","My brother");
$go1=switch($index2,"go","goes","goes");
$go2=switch($index2,"are going","is going","is going");
$ww1=switch($index1,"$go1","went","will go");
$ww2=switch($index1,"$go1","went","$go2");
$stad=switch(rint(3),"San Francisco","Rome", "Amsterdam");
```

In bovenstaand script worden eerst drie tijdsaanduidingen gedefinieerd.
Met de $index1 lopen de vervoegingen van het werkwoord "to go" in de pas met verleden, heden en toekomst en met $index2 lopen de verschillende persoonsvormen in de pas. In feite is hier sprake van nesting van de functie `switch`. De vraag is nu eenvoudig te stellen met:
$you (go) to $stad $tijdsbepaling.
En voor het antwoord gebruikt u weer het vraagtype *List* met twee mogelijke goede antwoorden $ww1 en $ww2 en gebruikmakend van een open invulveld. Zie ook *Figure 1.6 (page 4)*.

In het volgende voorbeeld moeten de twee verschillende antwoordvelden zelfs netjes in de pas lopen, wat in het *Algorithm* weer voorbereid wordt.

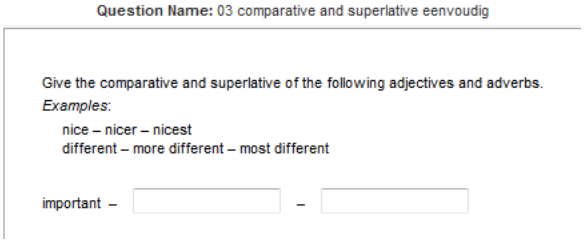


Figure 1.8: Antwoordvelden die in de pas moeten lopen

Hierboven is te zien dat de twee antwoordvelden met elkaar verband houden naar aanleiding van het gegeven.
Het bijbehorende *Algorithm* is hieronder te zien:

```
$index=rint(5);
$ad=switch($index,"bad","important","unhappy","far","serious");
$com1=switch($index,"worse","more important","unhappier","farther","more serious");
$com2=switch($index,"worse","more important","more unhappy","further","more serious");
$sup1=switch($index,"worst","most important","unhappiest","farthest","most serious");
$sup2=switch($index,"worst","most important","most unhappy","furthest","most serious");
```

We zien hier ook weer steeds twee antwoorden die soms verschillend en soms gelijk aan elkaar zijn.
Vergelijk ook de instellingen van het vraagtype *List* met *Figure 1.6 (page 4)*.

1.2.2 Multiple Choice

De tekst van een *Multiple Choice*-vraag kunt u ook wisselen op dezelfde manier als in bovenstaande voorbeelden. Wat de alternatieven (afleiders) van de *Multiple Choice*-vraag betreft, ondersteunt het programma zelf natuurlijk het doorelkaar aanbieden van de afleiders met het aanvinken van *Permuting*.
Er is echter nog meer mogelijk om de vraag steeds een ander aanzien te geven. U kunt bijvoorbeeld een hele serie foute antwoorden en een stuk of wat goede antwoorden formuleren en daar steeds een greep uit doen voor de uiteindelijke afleiders van de *Multiple Choice*-vraag. Al is het alleen maar dat de zinsbouw iets varieert om toch de vraag weer een hele andere uitstraling te geven.

Het is zelfs mogelijk om naar aanleiding van de steeds wisselende vraag ook een steeds ander correct antwoord aan te wijzen zoals te zien is in de volgende paragraaf.

1.2.2.1 Variabele als index voor de correcte afleider

In het volgende voorbeeld verandert steeds de vraag en verandert daarmee ook het correcte antwoord bij gelijkblijvende keuzemogelijkheden.

Figure 1.9: Variabele als correcte antwoord van meerkeuze

Het is mogelijk om bij het vraagtype *Multiple Choice* het goede antwoord te definiëren middels een variabele die in het *Algorithm* is voorbereid. Een dergelijk voorbeeld vindt u ook in paragraaf *Multiple Choice met berekeningen* *(page 36)*.

```
$index=rint(3);
$toren=switch($index,"Oldehove in Leeuwarden", "Martinitoren in Groningen","Euromast in Rotterdam");
$antw=switch($index,1,2,3);
```

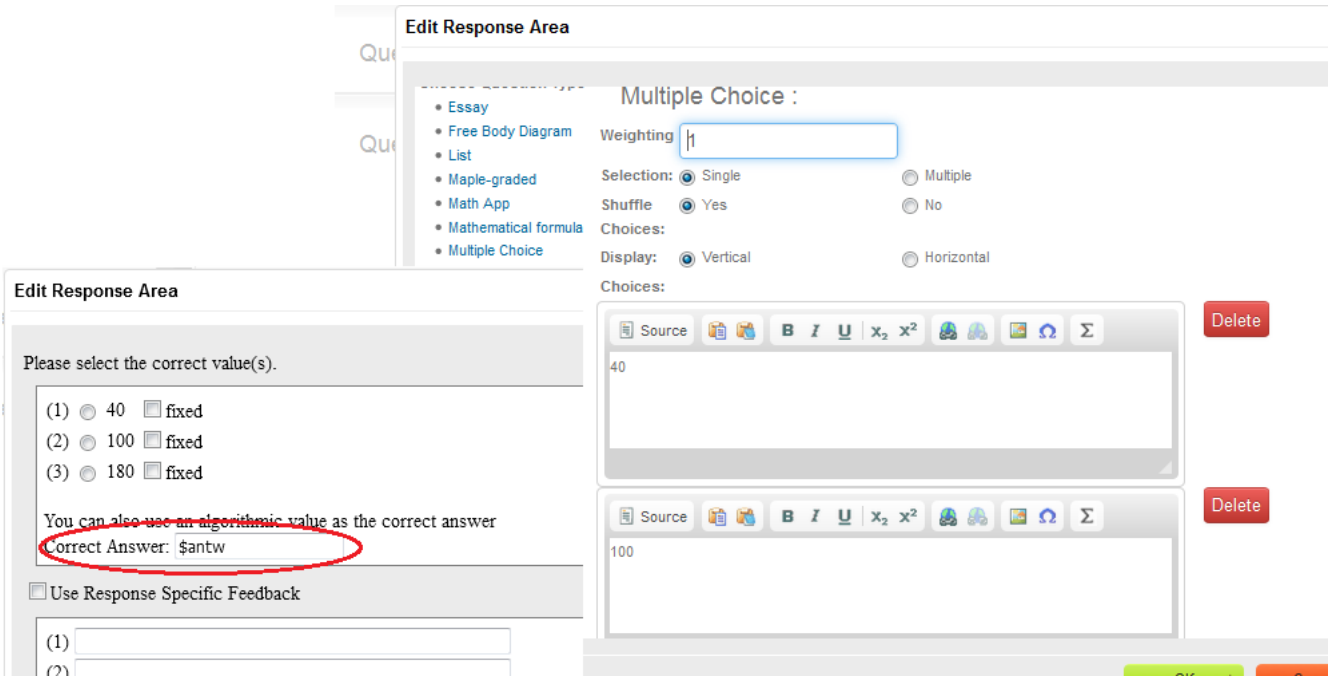


Figure 1.10: Algoritmische variabele als index voor correct antwoord

De alternatieven zijn steeds dezelfde (40, 100 en 180). Deze alternatieven worden doorelkaar (*Permuting*) aangeboden, maar afhankelijk van de vraag: "Wat is de hoogte van de $toren?" is het bijbehorende antwoord ($antw) in de vorm van een getal ook steeds verschillend. Dit getal geeft aan welk antwoord het correcte antwoord is (1) of (2) of (3).

TIP: Let erop dat als u gebruikmaakt van de *Aglorithmic Value* voor het aanwijzen van het juiste antwoord, dat u dan geen enkele radio button selecteert als zijnde het correcte antwoord.

1.2.2.2 Welk plaatje is het juiste?

Soms hebt u een *Multiple Choice*-vraag met een aantal plaatjes die u in een tabel netjes rangschikt.
Met een gewone Multiple Choice-vraag kunt u wel plaatjes invoeren als het kleine plaatjes zijn, maar als het er veel worden of als ze wat groot zijn, dan is het mooier om ze in een tabel te rangschikken en er A, B en C en dergelijke bij te schrijven.
De vraag is dan welk plaatje het juiste is. A, B, C of D enzovoort. De student kan dan in een *Multiple Choice*-vraag voor een van de letters opteren.

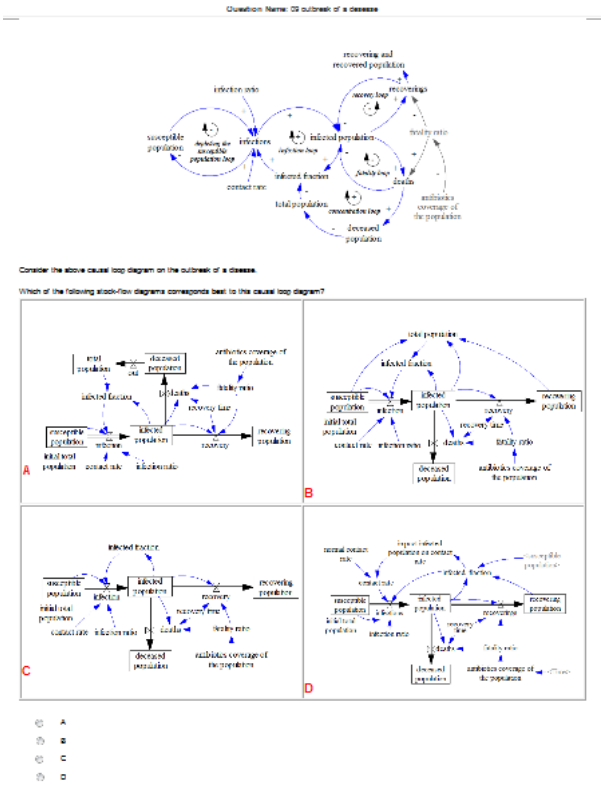


Figure 1.11: Plaatjes in een schema met daaronder een MC-vraag

Het mooiste is om de plaatjes steeds op willekeurige manier te rangschikken. Het juiste antwoord (A, B...) is dan ook steeds verschillend.
Stel dat u vier plaatjes hebt met verschillende namen en stel dat het plaatje met de naam "naam2" het juiste is.
Het *Algorithm* kan dan als volgt gemaakt worden:

```
$lijstA=maple("[0,1,2,3]");
$A=maple("randomize():StringTools[Randomize]():combinat[randperm]($lijstA)");

$index1=switch(0,$A);

$index2=switch(1,$A);

$index3=switch(2,$A);

$index4=switch(3,$A);
$pl1="naam1";
$pl2="naam2";
$pl3="naam3";
$pl4="naam4";
```

```
$a1=switch($index1,"$pl1","$pl2","$pl3","$pl4");
$a2=switch($index2,"$pl1","$pl2","$pl3","$pl4");
$a3=switch($index3,"$pl1","$pl2","$pl3","$pl4");
$a4=switch($index4,"$pl1","$pl2","$pl3","$pl4");
$antw=maple("ListTools[Search](1,$A)");
```

U hebt $lijstA waarvan het tweede getal (1) correspondeert met het tweede plaatje $pl2.
Deze lijst wordt nu doorelkaar gegooid tot een niewe lijst $A.

Met Maple zoeken we nu het getal (1) in deze lijst $A en kijken op welke plaats het staat en dan is dat het antwoord.
Het antwoord is dus een algoritmische variabele.

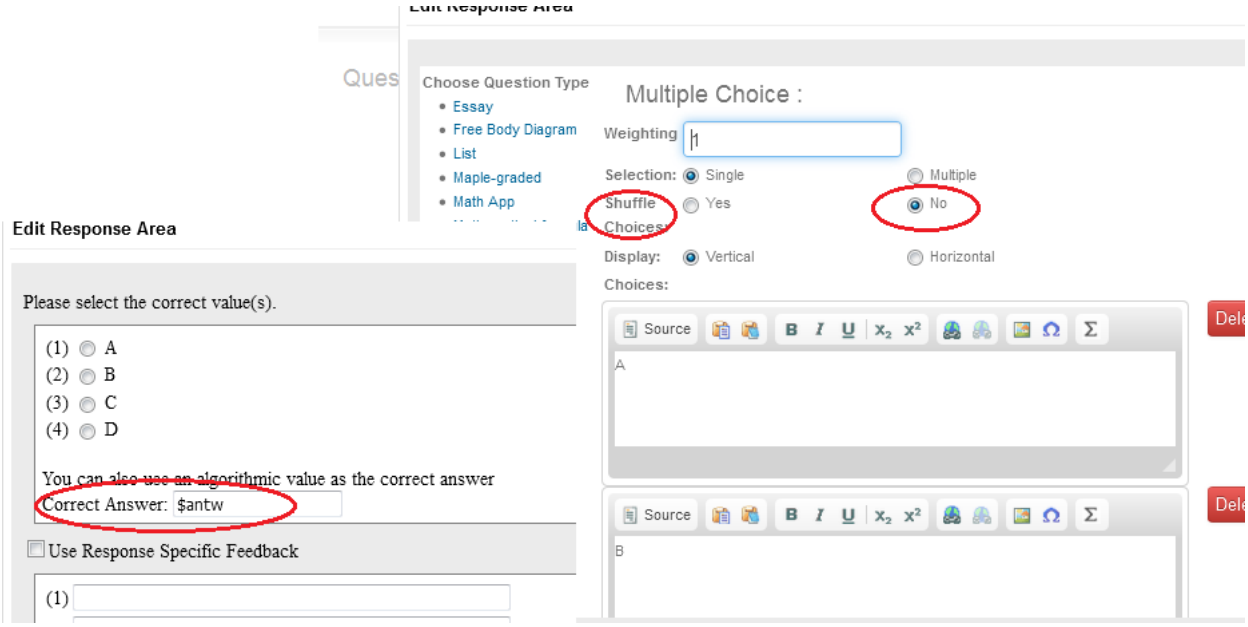


Figure 1.12: Een algoritmische variabele als antwoord van een MC-vraag

Stel dat er meer plaatjes juist zijn, dan wordt het iets lastiger.
Maak dan een extra lijst $B die correspondeert met $A. Stel dat plaatje $pl2 en plaatje $pl4 de juisten zijn.
Merk op dat $lijstA=[0,1,2,3] in de pas loopt met [0,1,0,1] (fout, goed, fout, goed).

```
$b1=switch($index1,0,1,0,1);
$b2=switch($index2,0,1,0,1);
$b3=switch($index3,0,1,0,1);
$b4=switch($index4,0,1,0,1);
$B=maple("[$b1,$b2,$b3,$b4]");
$antw=maple("ListTools[SearchAll](1,$B)");
```

In deze lijst laat u de getallen 1 zoeken met ListTools[SearchAll] en wat daar uit komt, is het antwoord en kan als rijtje ingevuld worden bij de *Multiple-Choice*-vraag in de rubriek *Algorithmic Value* als antwoord. Zet dan wel de instellingen op *Multiple*.

TIP: Merk op dat u natuurlijk kiest voor *Non Permuting* omdat anders A, B en C en D door elkaar gegooid zouden worden en dat moet natuurlijk hier niet.

1.2.2.3 Variëren van afleiders

In het volgende voorbeeld is steeds de vraag hetzelfde maar de alternatieven (afleiders) zijn steeds iets anders geformuleerd hoewel ze eigenlijk hetzelfde zijn wat betekenis betreft. Het programma kan met *Permuting* de alternatieven steeds op willekeurige volgorde aanbieden.

Figure 1.13: Verschillende formuleringen van de afleiders

In het *Algorithm* hiervan zien we dat de afleiders (één goed antwoord en drie foute antwoorden) steeds even iets anders geformuleerd worden. Willekeurig wordt er met switch een formulering gekozen.

$antwg=switch(rint(2),"Om de beschikbare grond zorgvuldig te verdelen.","Voor de zorgvuldige verdeling van de beschikbare grond.");
$antwf1=switch(rint(2),"Voor de verkoop van kavels.","Voor het verkopen van de kavels.");
$antwf2=switch(rint(2),"Om de veiligheid van bewoners te garanderen.","Voor het garanderen van de veiligheid van de bewoners.");
$antwf3=switch(rint(2),"Dit is een eis van de gemeente.","Dit wordt door de gemeente vereist.");

TIP: Als de alternatieven langer zijn, kunt u html-tags gebruiken voor de lay out.

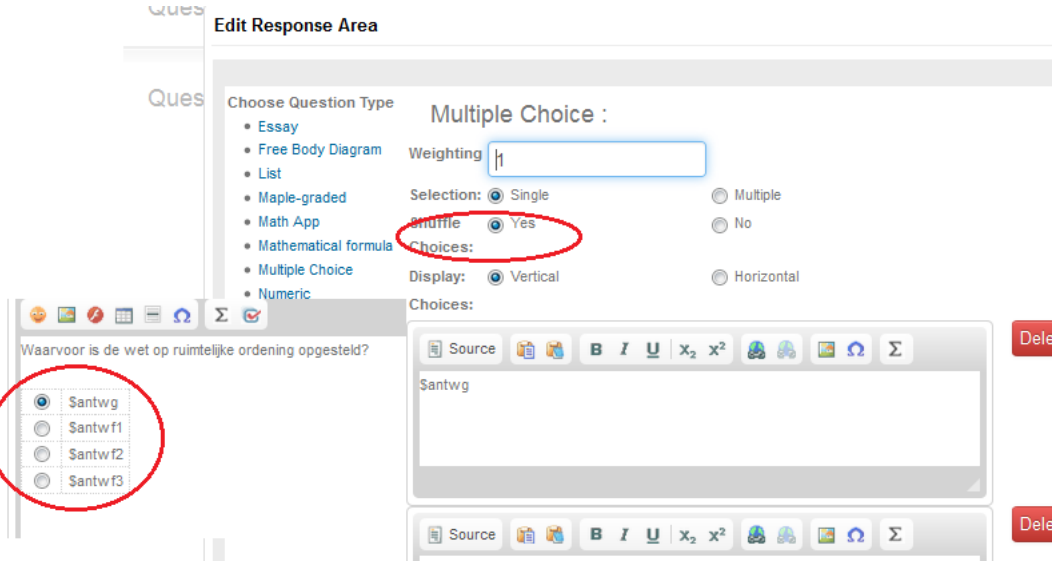


Figure 1.14: Verschillende formuleringen van de afleiders

In het volgende voorbeeld is steeds de vraag hetzelfde, maar er wordt willekeurig één goed antwoord gepresenteerd uit drie en er worden twee foute antwoorden willekeurig genomen uit vier. De alternatieven worden in het *Algorithm* aangemaakt en het programma kan de alternatieven uiteindelijk steeds doorelkaar aanbieden.

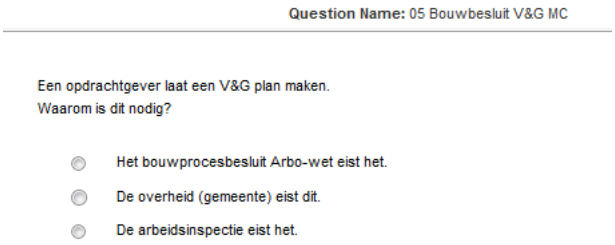


Figure 1.15: Multiple Choice met wisselende alternatieven

In het bijbehorende *Algorithm* worden er eerst drie goede antwoorden en 4 foute antwoorden geformuleerd.

```
$antwg1="De overheid (gemeente) eist dit.";
$antwg2="Dit is een eis van de gemeente (overheid).";
$antwg3="Dit is een eis van de overheid (gemeente).";
$antwf1="Een aannemer eist het.";
$antwf2="Het bouwprocesbesluit Arbo-wet eist het.";
$antwf3="Een onderaannemer eist het.";
$antwf4="De arbeidsinspectie eist het.";
$antwg=switch(rint(3),"$antwg1","$antwg2","$antwg3");
$A=maple("randomize():combinat[randcomb]([0,1,2,3],2)");

$index1=switch(0,$A);

$index2=switch(1,$A);
$antwF1=switch($index1,"$antwf1","$antwf2","$antwf3","$antwf4");
$antwF2=switch($index2,"$antwf1","$antwf2","$antwf3","$antwf4");
```

Met behulp van Maple wordt er een lijstje $A gegenereerd bestaande uit 2 elementen at random samengesteld uit de lijst [0,1,2,3]. We weten dan zeker dat deze twee elementen: de indices $index1 en $index2, niet gelijk zijn aan elkaar. Zie meer informatie over deze actie in paragraaf *Overzicht Permutaties (page 47)*.
Op deze manier kunnen we een greep maken van twee verschillende foute antwoorden uit de vier vooraf gedefinieerde foute antwoorden. Voor het vraagtype *Multiple Choice* binnen de *Question Designer* kunt u het goede antwoord en de twee foute antwoorden invoeren. Het systeem zal met *Permuting* de verschillende antwoorden doorelkaar aanbieden.
Zie ook *Figure 1.14 (page 9)*.

1.2.2.4 Variëren van tekst én afleiders

In het volgende voorbeeld is er steeds een andere formulering gekozen van de tekst van de vraag zodat de uitstraling van de vraag iedere keer net even anders is. Ook de afleiders van deze Multiple Choice-vraag zijn steeds anders geformuleerd. Hier zijn twee goede antwoorden waaruit één gekozen wordt en drie foute antwoorden voorbereid waaruit een greep van 2 genomen wordt.

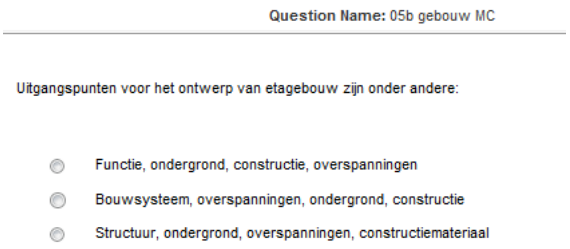


Figure 1.16: Vraag steeds anders en alternatieven steeds anders

Het *Algorithm* is als volgt gemaakt:

```
$gebouw=switch(rint(3),"stapelbouw","etagebouw","een gebouw met verdiepingen");
$antwg1="Structuur, ondergrond, overspanningen, constructiemateriaal";
$antwg2="Ondergrond, overspanningen, constructiemateriaal, structuur";
$antwg=switch(rint(2),"$antwg1","$antwg2");
$antwf1="Bouwsysteem, fundering, overspanningen, materialen";
$antwf2="Functie, ondergrond, constructie, overspanningen";
$antwf3="Bouwsysteem, overspanningen, ondergrond, constructie";
$A=maple("randomize():combinat[randcomb]([0,1,2],2)");

$index1=switch(0,$A);
```

```
$index2=switch(1,$A);
$antwF1=switch($index1,"$antwf1","$antwf2","$antwf3");
$antwF2=switch($index2,"$antwf1","$antwf2","$antwf3");
```

Vergelijk dit *Algorithm* ook eens met die van *Figure 1.15 (page 9)*.
De vraag heeft de formulering: "Uitgangspunten voor het ontwerp van een $gebouw zijn onder andere: "
en is daardoor steeds een beetje anders, hoewel er wel hetzelfde gevraagd wordt.

In het volgende voorbeeld is er van een opsomming sprake. De opsommingen kunnen mooi afzonderlijk doorelkaar aangeboden worden binnen de afleiders.

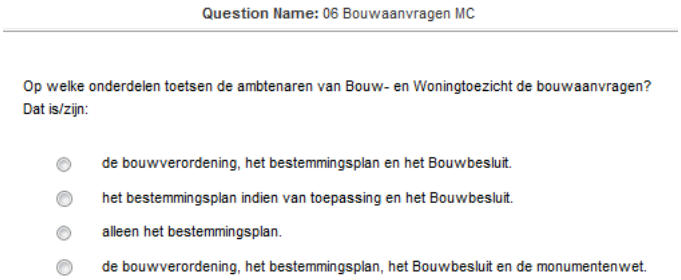


Figure 1.17: Opsommingen doorelkaar aanbieden

Het bijbehorende *Algorithm* laat zien hoe er verschillende indices worden aangemaakt.
Het nieuwe is hier dat de volgorde van de indices in het *Algorithm* op voorhand alvast door elkaar gegooid moet worden omdat er een opnoeming plaatsvindt in elke regel. Daarom nemen we een random permutatie van de lijst [1,2,3] ($A) voordat de indices kunnen worden aangemaakt. Net zo geldt dat voor de lijst $B. Zie ook in paragraaf *Overzicht Permutaties (page 47)* waarin meer uitleg staat over dit script.

```
$A=maple("randomize():StringTools[Randomize]():combinat[randperm]([0,1,2])");

$indexA1=switch(0,$A);

$indexA2=switch(1,$A);

$indexA3=switch(2,$A);
$B=maple("randomize():StringTools[Randomize]():combinat[randperm]([0,1,2,3])");
$indexB1=switch(0,$B);
$indexB2=switch(1,$B);
$indexB3=switch(2,$B);
$indexB4=switch(3,$B);
$a1="het bestemmingsplan";
$a2="de bouwverordening";
$a3="het Bouwbesluit";
$a4="de monumentenwet";
$Antwg1=switch($indexA1,"$a1","$a2","$a3");
$Antwg2=switch($indexA2,"$a1","$a2","$a3");
$Antwg3=switch($indexA3,"$a1","$a2","$a3");
$Antwf1=switch($indexB1,"$a1","$a2","$a3","$a4");
$Antwf2=switch($indexB2,"$a1","$a2","$a3","$a4");
$Antwf3=switch($indexB3,"$a1","$a2","$a3","$a4");
$Antwf4=switch($indexB4,"$a1","$a2","$a3","$a4");
```

Dit script kan eventueel vaker worden gebruikt als alleen de definities van de variabelen $a worden aangepast.

De antwoorden in de vorm van *Multiple Choice* kunnen er dan als volgt uitzien in de editor van de vraag:

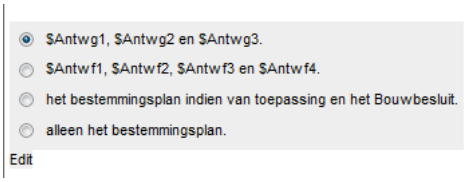


Figure 1.18: title of the figure

Als er gekozen wordt voor *Permuting*, worden de afleiders als geheel natuurlijk ook weer steeds door elkaar aangeboden.

1.2.2.5 Afleiders afhankelijk van de vraag

Bij de volgende vraag is het wel aardig dat in de vraag zelf geswitcht wordt tussen twee termen en bij de ene term zijn er 5 goede én 5 foute antwoorden gedefinieerd. Bij de andere term wordt er gewisseld tussen de goede en de foute antwoorden, dan is het in feite net andersom.

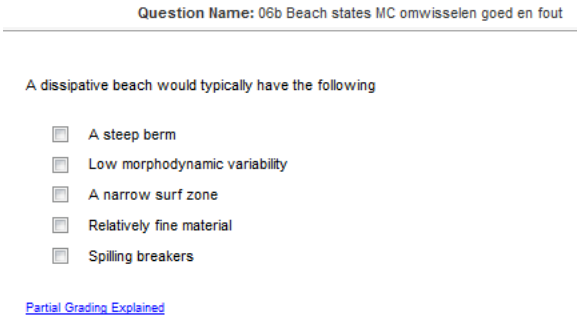


Figure 1.19: Omwisselen van goed en fout

Het bijbehorende *Algorithm* is hier onder te zien. Dit *Algorithm* is weer vrij robuust opgezet zodat het in meer situaties gebruikt kan worden. Er wordt in onderstaand script ook nog commentaar toegevoegd in de vorm van variabelen ($c1 en $c2).

$c1="Twee goede elementen kiezen uit 5 en drie foute elementen kiezen uit 5 andere. Maak alvast twee lijstjes met indices aan. Deze lijstjes hoeven niet gepermuteerd te worden, want de goede en foute antwoorden worden in de Multiple Choice vraag vanzelf wel gepermuteerd met de instelling *Permuting*.";
$A=maple("randomize():combinat[randcomb]([0,1,2,3,4],2)");
$indexg1=switch(0,$A);
$indexg2=switch(1,$A);
$B=maple("randomize():combinat[randcomb]([0,1,2,3,4],3)");
$indexf1=switch(0,$B);
$indexf2=switch(1,$B);
$indexf3=switch(2,$B);
$a1="A narrow surf zone";
$a2="Collapsing or surging breakers";
$a3="Relatively coarse material";
$a4="Low morphodynamic variability";
$a5="A steep berm";
$b1="A wide surf zone";
$b2="Spilling breakers";
$b3="Relatively fine material";
$b4="High morphodynamic variability";
$b5="Multiple linear bars";
$choice=rint(2);

```
$term =switch($choice,"reflective","dissipative");
$c2="Omdat er slechts twee mogelijkheden zijn voor $choice kan hier goed gebruikgemaakt worden van not($choice).";
$goed1 =
switch($choice,switch($indexg1,"$a1","$a2","$a3","$a4","$a5"),
switch($indexg1,"$b1","$b2","$b3","$b4","$b5"));
$goed2 =
switch($choice,switch($indexg2,"$a1","$a2","$a3","$a4","$a5"),
switch($indexg2,"$b1","$b2","$b3","$b4","$b5"));
$fout1 =
switch(not($choice),switch($indexf1,"$a1","$a2","$a3","$a4","$a5"),
switch($indexf1,"$b1","$b2","$b3","$b4","$b5"));
$fout2 =
switch(not($choice),switch($indexf2,"$a1","$a2","$a3","$a4","$a5"),
switch($indexf2,"$b1","$b2","$b3","$b4","$b5"));
$fout3 =
switch(not($choice),switch($indexf3,"$a1","$a2","$a3","$a4","$a5"),
switch($indexf3,"$b1","$b2","$b3","$b4","$b5"));
```

In bovenstaande script zijn twee commentaren ingevoegd $c1 en $c2.
Er worden uiteindelijk twee goede en drie foute antwoorden gegenereerd.
Dit is een mooi voorbeeld van het nesten van de functie switch.
Stel de index $choice is gelijk aan 0 ("reflective"), dan zijn de variabelen $a de goede antwoorden en als de index $choice gelijk is aan 1, dan zijn $b de goede antwoorden. Uit de 5 mogelijke goede antwoorden worden er willekeurig een greep van twee genomen en uit de 5 mogelijke foute antwoorden wordt er willekeurig een greep van 3 genomen. (Let ook even op de TIP aan het eind van dit voorbeeld omdat dit misschien niet wenselijk is voor specifiek deze situatie.)
Zie voor meer uitleg van bovenstaand script in paragraaf *Overzicht Permutaties (page 47)*.

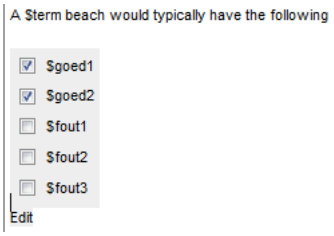


Figure 1.20: Nesten met behulp van switch

De variabele $term in de vraag, loopt dus in de pas met de voorbereide antwoorden.

TIP: Bij dit voorbeeld was het in feite nog beter geweest als de twee setjes indices $a en $b niet onafhankelijk van elkaar gekozen worden zoals hier. Immers als $indexg1 gelijk is aan 0 en $indexf1 ook gelijk is aan 0, dan krijgen we in ieder geval de twee alternatieven "A narrow surf zone" en "A wide surf zone" te zien en dat geeft iets van het antwoord weg. Beter is in deze situatie te programmeren:

```
$A=maple("StringTools[Randomize]():combinat[randperm]([0,1,2,3,4])");
$indexg1=switch(0,$A);
$indexg2=switch(1,$A);
$indexf1=switch(2,$A);
$indexf2=switch(3,$A);
$indexf3=switch(4,$A);
```

1.2.3 Stellingen juist of onjuist (multiple choice of drop-down menu)

Het komt vaak voor dat in een vraag een tweetal stellingen wordt aangeboden waarvan de student moet bepalen of deze wel of niet juist zijn. In het volgende voorbeeld kunt u de twee gegeven stellingen bijvoorbeeld niet alleen omwisselen, maar ook nét even iets anders formuleren. Voor de meerkeuzevraag zijn dan 4 afleiders te maken.

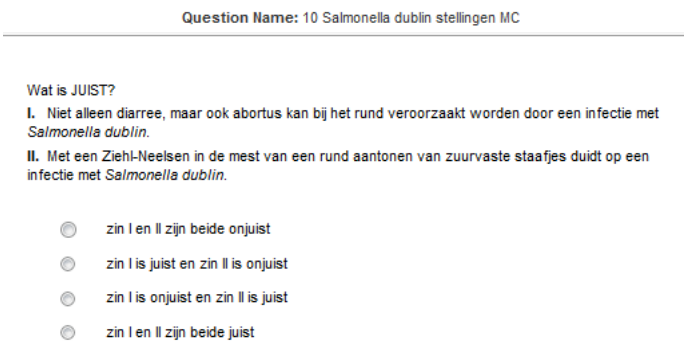


Figure 1.21: Stellingen juist of onjuist Multiple Choice

In het volgende *Algorithm* worden een onjuiste zin $zino en een juiste zin $zinj geformuleerd, elk willekeurig gekozen uit twee verschillende formuleringen.

```
$zinoa="Het aantonen van zuurvaste staafjes met een Ziehl-Neelsen in de mest van een rund wijst op een infectie met <i>Salmonella dublin</i>.";
$zinob="Met een Ziehl-Neelsen in de mest van een rund aantonen van zuurvaste staafjes duidt op een infectie met <i>Salmonella dublin</i>.";
$zino=switch(rint(2),"$zinoa","$zinob");
$zinja="Infecties met <i>Salmonella dublin</i> bij het rund leiden niet alleen tot diarree, maar kunnen ook abortus veroorzaken.";
$zinjb="Niet alleen diarree, maar ook abortus kan bij het rund veroorzaakt worden door een infectie met <i>Salmonella dublin</i>.";
$zinj=switch(rint(2),"$zinja","$zinjb");
$index1=rint(2);
$index2=not($index1);
$ZIN1=switch($index1,"$zino","$zinj");
$ZIN2=switch($index2,"$zino","$zinj");
$antwg=if(lt($index1,$index2),"zin I is onjuist en zin II is juist","zin I is juist en zin II is onjuist");
$antwf1=if(lt($index1,$index2),"zin I is juist en zin II is onjuist","zin I is onjuist en zin II is juist");
$antwf2="zin I en II zijn beide juist";
$antwf3="zin I en II zijn beide onjuist";
```

Hierin heeft "if(lt($index1,$index2)" de betekenis van "als $index1 kleiner is dan $index2 dan ... anders....". Zie ook paragraaf *Randomgetallen (page 54)*.
Voor meer uitleg over bovenstaand script zie paragraaf *Overzicht Permutaties (page 47)*.

De vraag ziet er dan als volgt uit in de editor en deze vraag zou met een kleine aanpassing als template kunnen dienen voor dit soort vragen met twee stellingen. Alleen de formuleringen van de stellingen hoeft maar aangepast te worden.

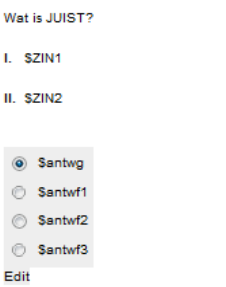


Figure 1.22: Stellingen juist of onjuist in het vraagtype Multiple Choice

Dus niet alleen de zinnen worden omgewisseld maar ook steeds iets anders geformuleerd.

Hoewel dit soort vragen vaak voorkomen, zijn dit niet altijd de beste vragen! Er zijn andere manieren om stellingen te overhoren zoals in de volgende voorbeelden te zien is.

Een andere manier is de student te laten aanvinken in een *Multiple Choice*-vraag welke stelling van meer aangeboden stellingen juist is. U kunt er dan ook voor kiezen dat er meer stellingen juist kunnen zijn (*Multiple*). Bereid in het *Algorithm* dan steeds verschillende formuleringen van de stellingen voor, zodat de vraag steeds een andere uitstraling krijgt. Ook de formulering van de vraag zelf kunt u steeds wisselen.

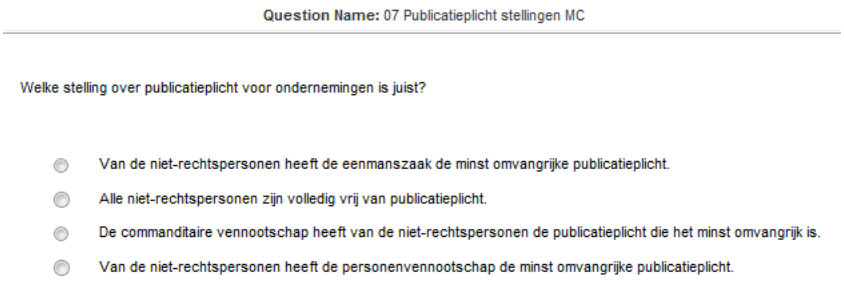


Figure 1.23: Stellingen juist Multiple Choice of Multiple Answer

Het bijbehorende *Algorithm* is vrij overzichtelijk. Let op dat hier gebruikgemaakt is van html-tags voor de lay-out van de vraag die op 3 verschillende manieren geformuleerd kan worden.

```
$vraag1="De publicatieplicht is niet voor alle ondernemingen gelijk.<br><br>Welke stelling hierover is juist?";
$vraag2="Voor alle ondernemingen is de publicatieplicht niet gelijk.<br><br>Geeft de juiste stelling hierover aan.";
$vraag3="Welke stelling over publicatieplicht voor ondernemingen is juist?";
$vraag=switch(rint(3),"$vraag1","$vraag2","$vraag3");
$antwg1="Alle niet-rechtspersonen zijn volledig vrij van publicatieplicht.";
$antwg2="Volledig vrij van publicatieplicht zijn alle niet-rechtspersonen.";
$antwf11="Van de niet-rechtspersonen heeft de eenmanszaak de minst omvangrijke publicatieplicht.";
$antwf12="De eenmanszaak heeft van de niet-rechtspersonen de publicatieplicht die het minst omvangrijk is.";
$antwf21="Van de niet-rechtspersonen heeft de personenvennootschap de minst omvangrijke publicatieplicht.";
$antwf22="De personenvennootschap heeft van de niet-rechtspersonen de publicatieplicht die het minst omvangrijk is.";
$antwf31="Van de niet-rechtspersonen heeft de commanditaire vennootschap de minst omvangrijke publicatieplicht.";
$antwf32="De commanditaire vennootschap heeft van de niet-rechtspersonen de publicatieplicht die het minst omvangrijk is.";
$antwg=switch(rint(2),"$antwg1","$antwg2");
$antwf1=switch(rint(2),"$antwf11","$antwf12");
$antwf2=switch(rint(2),"$antwf21","$antwf22");
$antwf3=switch(rint(2),"$antwf31","$antwf32");
```

Als u meer dan één goed antwoord voorbereid, dan kunt u met dit vraagtype opteren voor *Multiple*. De radio buttons veranderen dan automatisch in checkboxes.

Nog een mogelijkheid is om achter alle stellingen een drop-down menu aan te bieden met de keuze juist of onjuist. Elke stelling wordt dan apart beoordeeld. (In dit voorbeeld is er ook nog een *Essay*-vraag aan toegevoegd.)

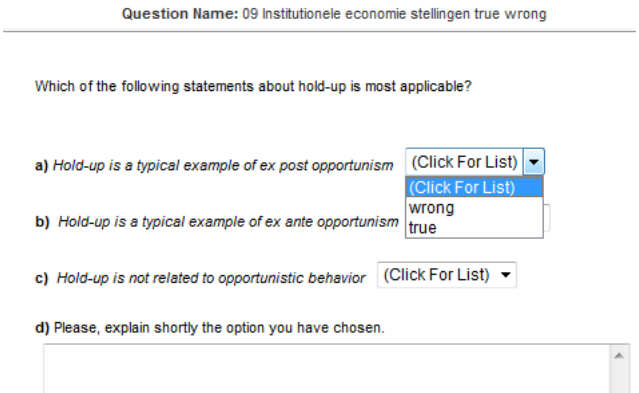


Figure 1.24: Stellingen juist of onjuist met drop-down menu

Deze manier van doen is in feite meer een soort matchingsvraag. Voordeel hiervan is ook dat de student deelpunten kan scoren op deze vraag.
Het programma biedt hier de verschillende stellingen echter niet doorelkaar. Als u dat wel wilt, moet u zelf in het *Algorithm* de volgorde van de verschillende stellingen alvast willekeurig maken met daarbij in de pas lopend natuurlijk ook de juiste antwoorden!
Het volgende *Algorithm* hoort hier dan ook bij:

```
$st1="Hold-up is a typical example of ex post opportunism";
$st2="Hold-up is a typical example of ex ante opportunism";
$st3="Hold-up is not related to opportunistic behavior";
$A=maple("StringTools[Randomize]():combinat[randperm]([0,1,2])");
$indexA1=switch(0,$A);
$indexA2=switch(1,$A);
$indexA3=switch(2,$A);
$stelling1=switch($indexA1,"$st1","$st2","$st3");
$stelling2=switch($indexA2,"$st1","$st2","$st3");
$stelling3=switch($indexA3,"$st1","$st2","$st3");
$antw1g=switch($indexA1,"true","wrong","wrong");
$antw1f=switch($indexA1,"wrong","true","true");
$antw2g=switch($indexA2,"true","wrong","wrong");
$antw2f=switch($indexA2,"wrong","true","true");
$antw3g=switch($indexA3,"true","wrong","wrong");
$antw3f=switch($indexA3,"wrong","true","true");
```

Voor elke stelling wordt een goed en een fout antwoord voorbereid dat u in het vraagtype *List* kunt gebruiken.
Voor meer informatie over bovenstaand script met de permutaties van een lijst, zie paragraaf *Overzicht Permutaties* *(page 47)*.

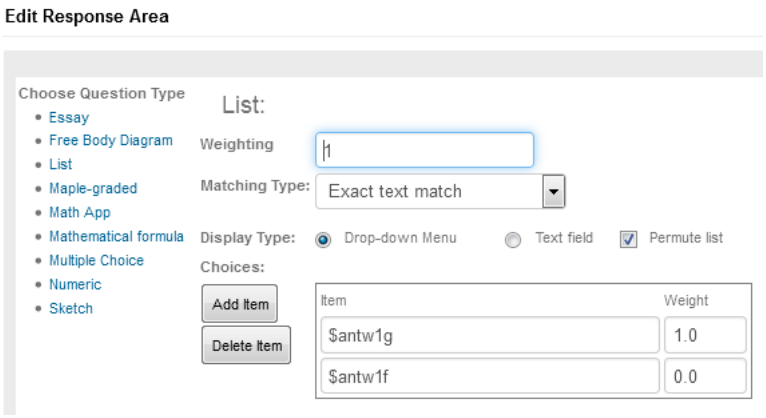


Figure 1.25: Stellingen beoordelen met true en wrong

1.2.4 Drop-down menu's (matching)

Het vraagtype *Matching* komt helaas niet voor binnen de *Question Designer*. Dit losse vraagtype is ook sterk beperkt. Er kan namelijk alleen maar bij elk gegeven één bijpassend antwoord horen. Dus *evenveel* gegevens en *evenveel* bijpassende antwoorden zijn slechts mogelijk. Ook kan er later geen invulveld meer aan toegevoegd worden zoals men bij de *Question Designer* wel kan doen. Ook met het oog op de adaptieve mogelijkheden van de *Adaptive Question Designer* is het belangrijk dat u matchingsvragen zoveel mogelijk maakt in de *Question Designer*.
Bij het losse vraagtype *Matching* wordt echter wel alles steeds automatisch door elkaar gehusseld en deze vraag is gemakkelijk en snel te maken. Verder kunt u in het *Algorithm* altijd een aantal dingen variëren.

1.2.4.1 Vraagtype Matching met tekstrandomisering

We geven eerst een voorbeeld van bij ieder gegeven één bijpassend antwoord. (Zie ook in de *Handleiding Möbius Items maken Deel A* bij vraagtype *Matching*.)

In het volgende voorbeeld wordt dus het zelfstandige vraagtype *Matching* gebruikt en moeten voorbeelden aan de verschillende begrippen van de beschrijvende statistiek gekoppeld worden.

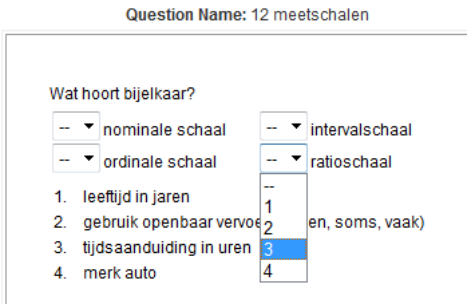


Figure 1.26: Matchingsvraag met randomisering

De bedoeling is dus om bepaalde begrippen met elkaar te koppelen.
Na openen van de vraag, komen we in het overzicht waar u de rubriek *Algorithm* kunt invullen ter voorbereiding van de variabelen *Figure 1.27 (page 17)*.

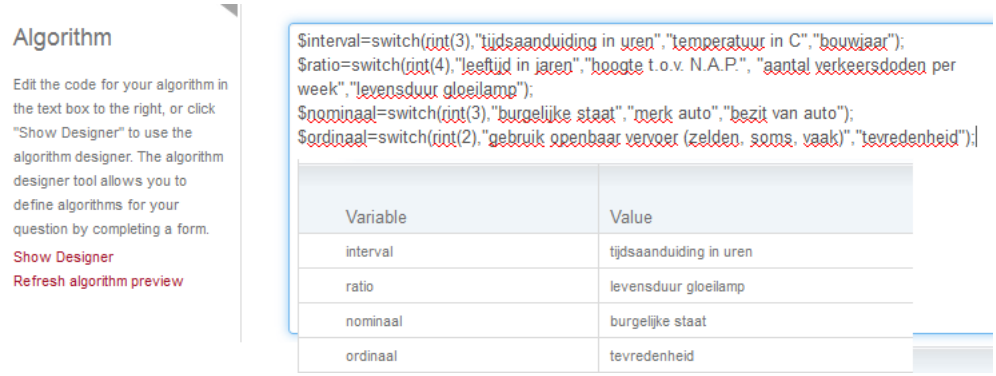


Figure 1.27: Het algoritme om een matchingsvraag te randomiseren

In de rubriek *Algorithm* is het volgende ingevoerd:

```
$interval=switch(rint(3),"tijdsaanduiding in uren","temperatuur in C","bouwjaar");
$ratio=switch(rint(4),"leeftijd in jaren","hoogte t.o.v. N.A.P.", "aantal verkeersdoden per week","levensduur gloeilamp");
$nominaal=switch(rint(3),"burgelijke staat","merk auto","bezit van auto");
$ordinaal=switch(rint(2),"gebruik openbaar vervoer (zelden, soms, vaak)","tevredenheid");
```

Deze variabelen worden in de te matchen items van de vraag aangeroepen en zo onstaan dan steeds andere voorbeelden van de verschillende schalen.
De functie rint(4) betekent 0 of 1 of 2 of 3 en geeft dus vier mogelijkheden waartussen geswitcht kan worden met de functie *switch*.

Bij de rubriek *Question Text* komt u in een formulier *Figure 1.28 (page 18)* waar de informatie over de vraag geformuleerd kan worden. Bij de rubriek *Matching* kunt u de onderdelen die bij elkaar moeten passen voorbereiden.

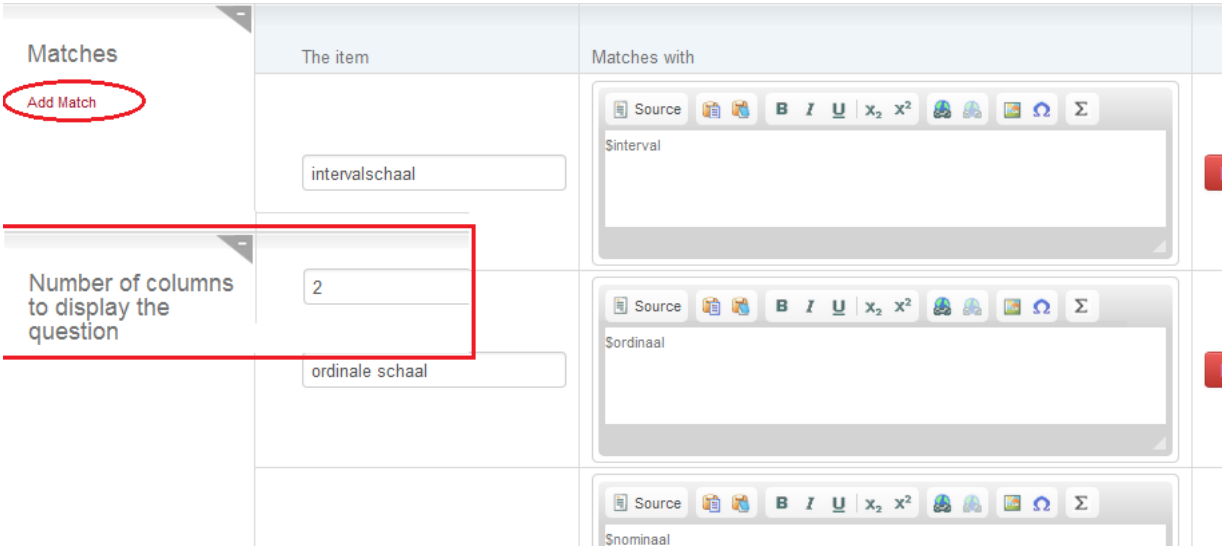


Figure 1.28: Editen van een Matchingsvraag

In bovenstaande *Figure 1.28 (page 18)* is te zien dat onder het tekstveld voor de vraag de matchingsonderdelen paarsgewijs ingevoerd kunnen worden (met *Add Match* links bovenaan), eventueel met plaatjes en gebruikmakend van de randomvariabelen die vooraf in de rubriek *Algorithm* zijn voorbereid.
Hier is bijvoorbeeld te zien dat het begrip "intervalschaal" moet matchen met de variabele $interval. Deze variabele staat dan voor de verschillende mogelijkheden die in de rubriek *Algorithm* zijn voorgeprogrammeerd, maar u kunt natuurlijk ook direct iets invullen al of niet met gebruikmaking van html-tags.
Nadat u iets ingevuld hebt, klikt u op *Add* links bovenaan en kunt u vervolgens eventueel weer iets nieuws invullen.
Met number of columns in which to display the question kunt u de items in meer kolommen aanbieden zoals hier te zien is in *Figure 1.26 (page 17)* in twee kolommen. Op die manier kunt u toch de lay-out iets beinvloeden.

TIP: Het is slechts mogelijk om *evenveel* alternatieven van de ene soort met *evenveel* alternatieven van de bijpassende soort te koppelen in een zelfstandige vraag van het type *Matching*. In de *Question Designer* zijn er meer mogelijkheden.

1.2.4.2 Vraagtype Matching met plaatjes

In het volgende voorbeeld wordt er een greep van vier plaatjes genomen uit in totaal 11 plaatjes met in de pas lopende antwoorden.

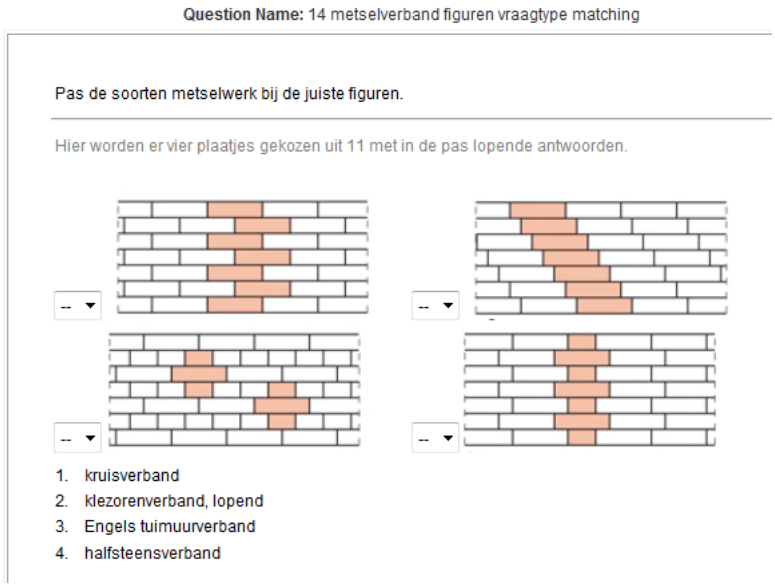


Figure 1.29: Matching met gerandomiseerde plaatjes en bijbehorende antwoorden

Eerst moeten de plaatjes geüpload worden in de *Website Editor* van de Class. Deze plaatjes hebben genummerde namen: "verband1.jpg" enz.
In het *Algorithm* is nu het volgende geprogrammeerd inclusief commentaar $c0, $c1 t/m $4, waarbij het script voor dit soort vragen helemaal overgenomen kan worden. Voor uw eigen vraag volstaat slechts een kleine aanpassing.

```
$c0="Er wordt een lijst gegenereerd van 4 getallen uit 11. Deze verkregen lijst is altijd op volgorde van grootte. Dat geeft hier niet, want het vraagtype Matching zorgt wel voor het doorelkaar gooien van de volgorde.";
$A=maple("randomize():combinat[randcomb]([0,1,2,3,4,5,6,7,8,9,10],4)");

$index1=switch(0,$A);

$index2=switch(1,$A);

$index3=switch(2,$A);

$index4=switch(3,$A);
$c1="Dan volgen nu de namen van de plaatjes. ";
$a1="verband1";
$a2="verband2";
$a3="verband3";
$a4="verband4";
$a5="verband5";
$a6="verband6";
$a7="verband7";
$a8="verband8";
$a9="verband9";
$a10="verband10";
$a11="verband11";
$c2="De volgende variabelen zijn de betekenissen van de plaatjes in de pas lopend met de namen van de plaatjes.";
$b1="halfsteensverband";
$b2="staand verband";
$b3="kruisverband";
$b4="klezorenverband, lopend";
$b5="klezorenverband, staand";
$b6="vrij of wild verband";
$b7="Noors of kettingverband";
$b8="Vlaams verband";
$b9="Frans verband";
$b10="Engels tuimuurverband";
$b11="koppenverband";
$c3="Er worden vier plaatjes uit 11 gekozen middels de index. ";
$pl1=switch($index1,"$a1","$a2","$a3","$a4","$a5","$a6","$a7","$a8","$a9","$a10","$a11");
$pl2=switch($index2,"$a1","$a2","$a3","$a4","$a5","$a6","$a7","$a8","$a9","$a10","$a11");
$pl3=switch($index3,"$a1","$a2","$a3","$a4","$a5","$a6","$a7","$a8","$a9","$a10","$a11");
$pl4=switch($index4,"$a1","$a2","$a3","$a4","$a5","$a6","$a7","$a8","$a9","$a10","$a11");
$c4="De antwoorden lopen in de pas middels de vooraf gedefinieerde indices";
$antw1=switch($index1,"$b1","$b2","$b3","$b4","$b5","$b6","$b7","$b8","$b9","$b10","$b11");
$antw2=switch($index2,"$b1","$b2","$b3","$b4","$b5","$b6","$b7","$b8","$b9","$b10","$b11");
$antw3=switch($index3,"$b1","$b2","$b3","$b4","$b5","$b6","$b7","$b8","$b9","$b10","$b11");
$antw4=switch($index4,"$b1","$b2","$b3","$b4","$b5","$b6","$b7","$b8","$b9","$b10","$b11");
```

Een plaatje invoegen is tegenwoordig zeer eenvoudig.

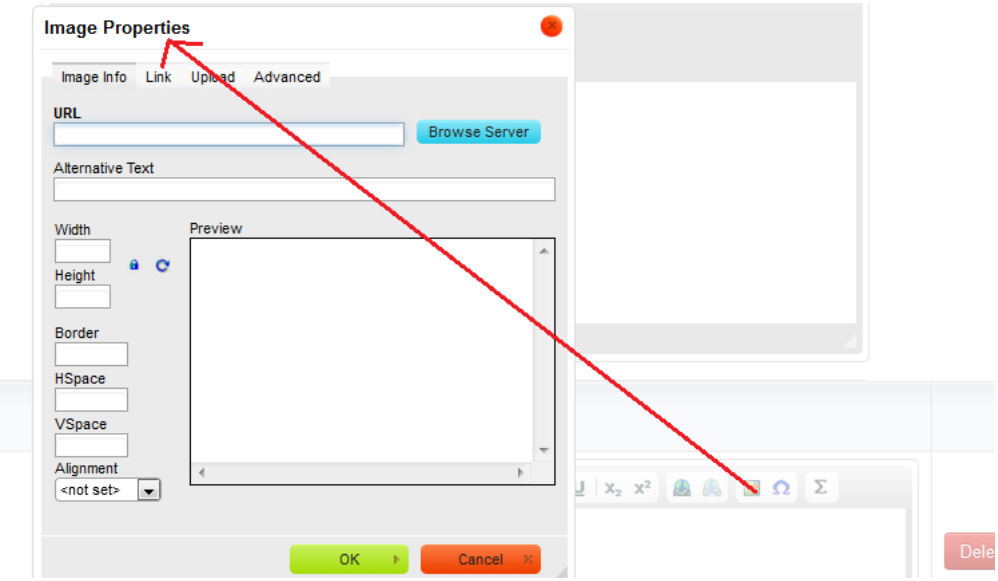


Figure 1.30: De url van een plaatje kopiëren

Vervolgens vervangt u de naam van het plaatje door de variabele $pl1 en dat plaatje moet natuurlijk matchen met het bijbehorende antwoord $antw1.

De rest van de items maakt u op dezelfde manier steeds met het aanpassen van de variabele naam van het plaatje.

De urls van de plaatjes kunt u ook kopieren naar de kolom met items en dat moet dan matchen met de onderschriften van de plaatjes bijvoorbeeld.

Vergeet niet de tag rond de url: <img src=".....url plaatje">.

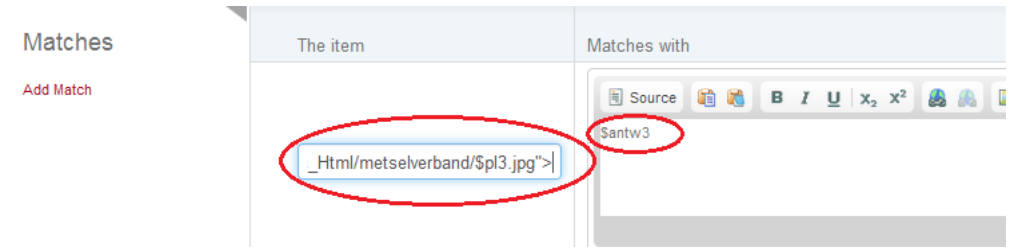


Figure 1.31: Plaatjes in de Matchingvraag

TIP: Zorg ervoor dat in de naam van het plaatje het antwoord niet verscholen ligt.

1.2.4.3 Vraagtype List in Question Designer (drop-down menu)

Als u de matchingsvraag zelf maakt in de *Question Designer*, hebt u in feite veel meer mogelijkheden:
Ten eerste hebt u meer invloed op de lay out.
Ten tweede hoeft u niet evenveel items en evenveel antwoorden te hebben.
Ten derde hoeven nu niet alle drop-down menu's hetzelfde te zijn.
Ten vierde kunnen verschillende velden verschillend gewaardeerd worden.
Ten vijfde kunt u uitbreiden naar het adaptieve vraagtype of nog antwoordvelden erbij maken.
Wat u echter nu wel moet doen is in het *Algorithm* alvast de volgorde door elkaar gooien.

In het volgende voorbeeld worden vier namen van onderdelen gegeven en de student moet de bijpassende letter erbij aangeven.
Er worden steeds 4 namen gegeven en 5 letters aangeboden.

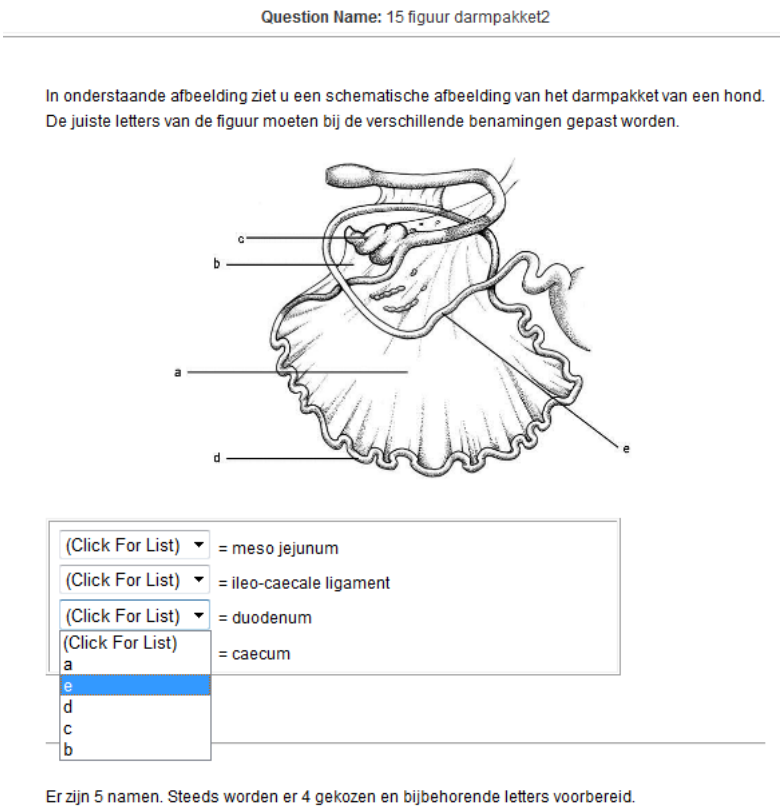


Figure 1.32: Namen moeten corresponderen met de letters in het plaatje

Het bijbehorende *Algorithm* staat hieronder:

```
$A=maple("randomize():StringTools[Randomize]():combinat[randperm]([0,1,2,3,4])");
$c1="De volgorde moet nu van te voren door elkaar gegooid zijn.";

$index1=switch(0,$A);

$index2=switch(1,$A);

$index3=switch(2,$A);

$index4=switch(3,$A);

$index5=switch(4,$A);
$a1="meso jejunum";
$a2="ileo-caecale ligament";
$a3="caecum";
$a4="jejunum";
$a5="duodenum";
$naam1=switch($index1,"$a1","$a2","$a3","$a4","$a5");
$naam2=switch($index2,"$a1","$a2","$a3","$a4","$a5");
$naam3=switch($index3,"$a1","$a2","$a3","$a4","$a5");
$naam4=switch($index4,"$a1","$a2","$a3","$a4","$a5");
$letter1=switch($index1,"a","b","c","d","e");
$letter2=switch($index2,"a","b","c","d","e");
$letter3=switch($index3,"a","b","c","d","e");
$letter4=switch($index4,"a","b","c","d","e");
$letter5=switch($index5,"a","b","c","d","e");
```

Opvallend is dat hier slechts 4 namen voorbereid zijn en er zijn 5 letters waaruit de student kan kiezen. Dat kunt u zelf variëren.

Bij het maken van het drop-down menu, kiest u voor vraagtype *List*.

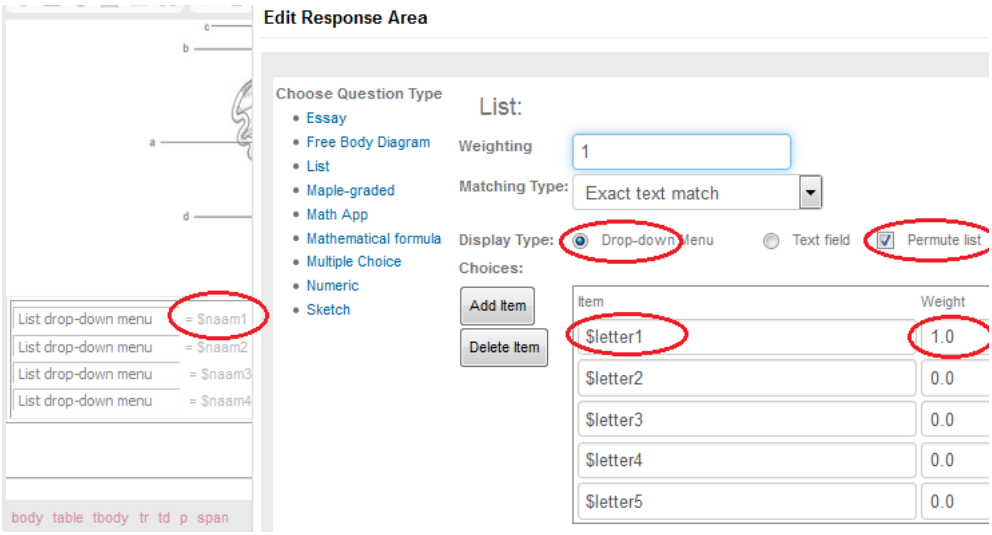


Figure 1.33: Drop-down menu maken

Laat bij het maken van de drop-down menu's steeds de naam corresponderen met het juiste antwoord. U kunt *Permute list* aanvinken zodat bij het drop-down menu steeds een andere volgorde aangeboden wordt.

In het volgende voorbeeld zijn er 4 termen gekozen uit 9. In de drop-down menu's staan steeds 5 mogelijke antwoorden waaronder één goede. Het aantal mogelijke antwoorden is dus kleiner dan het totaal aantal items.

Figure 1.34: Meer items en minder antwoorden met drop down menu

Het bijbehorende *Algorithm* is als volgt gemaakt waarin de lijst $AAA in de pas loopt met de lijst $B

```
$AAA=maple("[0,1,2,3,4,5,6,7,8,9]");
$AA=maple("randomize():combinat[randcomb]($AAA,4)");
$A=maple("StringTools[Randomize]():combinat[randperm]($AA)");
$B=maple("[0,0,1,1,2,2,3,4,4,4]");
$a1="aardbei";
$a2="bloed";
$a3="blank papier";
$a4="tanden";
```

```
$a5="wolkloze lucht";
$a6="Middellandse zee";
$a7="boterbloem";
$a8="gras";
$a9="komkommer";
$a10="stoplicht waarbij je mag doorrijden";
$b1="rood";
$b2="wit";
$b3="blauw";
$b4="geel";
$b5="groen";
$index1=switch(0,$A);
$index1g=switch($index1+1,$B);
$index2=switch(1,$A);
$index2g=switch($index2+1,$B);
$index3=switch(2,$A);
$index3g=switch($index3+1,$B);
$index4=switch(3,$A);
$index4g=switch($index4+1,$B);
$term1=switch($index1,"$a1","$a2","$a3","$a4","$a5","$a6","$a7","$a8","$a9","$a10");
$deel1g=switch($index1g,"$b1","$b2","$b3","$b4","$b5","$b1","$b2","$b3","$b4","$b5");
$deel1f1=switch($index1g+1,"$b1","$b2","$b3","$b4","$b5","$b1","$b2","$b3","$b4","$b5");
$deel1f2=switch($index1g+2,"$b1","$b2","$b3","$b4","$b5","$b1","$b2","$b3","$b4","$b5");
$deel1f3=switch($index1g+3,"$b1","$b2","$b3","$b4","$b5","$b1","$b2","$b3","$b4","$b5");
$deel1f4=switch($index1g+4,"$b1","$b2","$b3","$b4","$b5","$b1","$b2","$b3","$b4","$b5");
```

Zo ook voor $term2 en $deel2g en $deel2f1 enz.....

Let hierbij op dat bijvoorbeeld $index1=switch(0,$A); koppelt met het goede antwoord met index $index1g=switch($index1+1,$B); Als bijvoorbeeld $index1 gelijk is aan 3 (dan gaat het dus over $a4="tanden"), dan moeten we het vierde element hebben in de bijbehorende lijst $B= [0,0,1,1,2,2,3,4,4,4] en dat is dus het getal 1. (Immers Maple begint altijd te tellen bij 1 en bij switch begint het systeem te tellen bij 0). Dit element van de lijst $B koppelt dan met $b2="wit".
De waarden $b2, $b3 enzovoort zijn dus foute antwoorden die u krijgt door steeds de index met 1 op te hogen.
Op deze manier kunt u meer items $a koppelen met minder items $b.
In de drop-down menu's komen dan bijvoorbeeld bij $term2 de bijbehorende alternatieven $deel 2g als goed antwoord en $deel2f1 en $deel2f2 enz. als foute mogelijkheden.

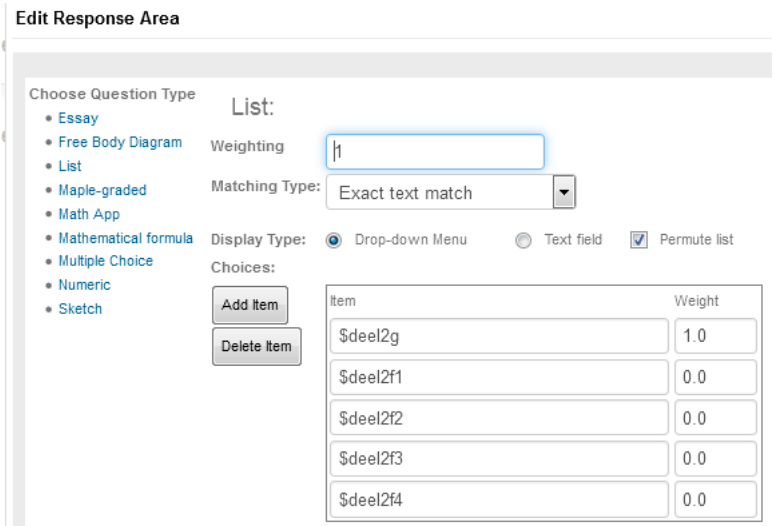


Figure 1.35: Drop-down menu met meer items en minder antwoorden

1.2.4.4 Drop-down menu's bij gerandomiseerde plaatjes

In het volgende voorbeeld zijn er slechts drie plaatjes beschikbaar en die worden steeds in wisselende volgorde geplaatst (in een tabel). Daaronder komen de drop-down menu's te staan met 6 namen waaruit gekozen kan worden.

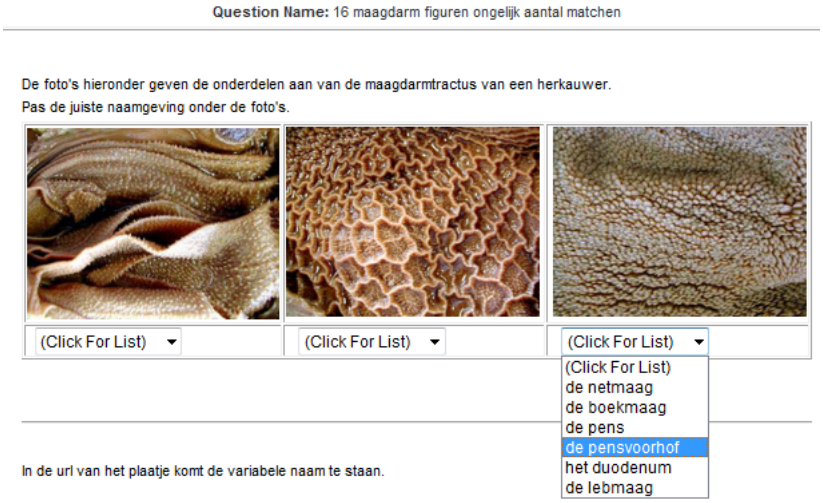


Figure 1.36: Drop-down menu van 3 plaatjes met 6 namen

Het bijbehorende *Algorithm* is:

```
$AA=maple("[0,1,2]");
$A=maple("randomize():StringTools[Randomize]():combinat[randperm]($AA)");
$c="De volgorde moet hier doorelkaar gegooid worden.";

$index1=switch(0,$A);

$index2=switch(1,$A);

$index3=switch(2,$A);
$a1="maagdarm1";
$a2="maagdarm2";
$a3="maagdarm3";
$b1="de pens";
$b2="de lebmaag";
$b3="de netmaag";
$foto1=switch($index1,"$a1","$a2","$a3");
$foto2=switch($index2,"$a1","$a2","$a3");
$foto3=switch($index3,"$a1","$a2","$a3");
$naam1=switch($index1,"$b1","$b2","$b3");
$naam2=switch($index2,"$b1","$b2","$b3");
$naam3=switch($index3,"$b1","$b2","$b3");
$naam4="de pensvoorhof";
$naam5="het duodenum";
$naam6="de boekmaag";
```

Er zijn dus nog drie namen die er niet toe doen aan toegevoegd.

TIP: Zorg ervoor dat de plaatjes wat betreft de afmetingen precies in de tabel passen voor een mooie lay out.

TIP: Zorg ervoor dat u namen van de plaatjes op geen enkele manier in verband brengt met het antwoord.

In het volgende voorbeeld maakt u zelf de matchingsvraag van bijvoorbeeld drie plaatjes (een greep uit 11 plaatjes) met een drop-down menu van 7 namen (een greep uit 11 namen) onder elk plaatje.

Deze uitvoering is in feite nog mooier en beter dan de matchingsvraag van *Figure 1.29 (page 18)*. Wel hoort er bij elk plaatje steeds maar één antwoord.

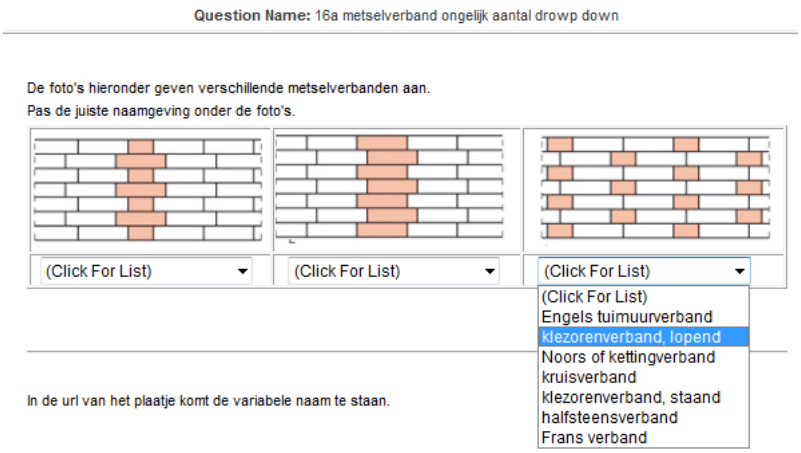


Figure 1.37: Matchting met de Question Designer

Vergelijk ook het *Algorithm* met dat van *Figure 1.29 (page 18)* en dat van *Figure 1.36 (page 24)*.

TIP: Zie in paragraaf *Overzicht Permutaties (page 47)* als u grotere aantallen namen of nummers van volgorde wilt laten wisselen.

1.2.4.5 Drop-down menu's bij een labeled image

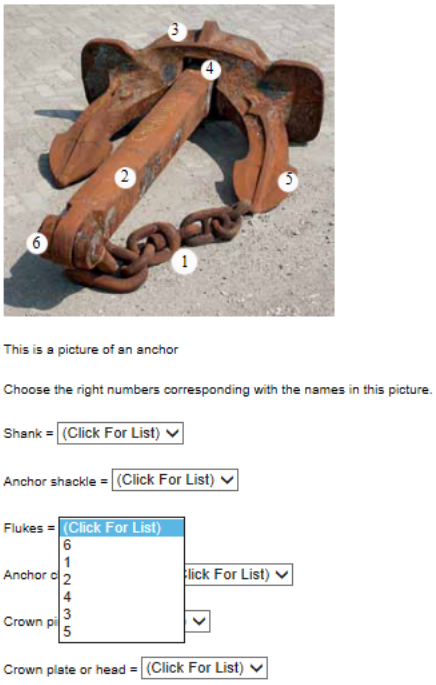


Figure 1.38: Labelled Image gerandomiseerd

Hier worden steeds andere namen genoemd $naam en de student moet met het drop-down-menu het juiste nummer kiezen.

Het bijbehorende *Algorithm* is:

$aa=range(1,1000);

```
$bb=range(1,1000);
$A=maple("StringTools[Randomize]($aa):combinat[randperm]([0,1,2,3,4,5])");

$indexA1=switch(0,$A);

$indexA2=switch(1,$A);

$indexA3=switch(2,$A);

$indexA4=switch(3,$A);

$indexA5=switch(4,$A);

$indexA6=switch(5,$A);
$B=maple("StringTools[Randomize]($bb):combinat[randperm]([0,1,2,3,4,5])");

$indexB1=switch(0,$B);

$indexB2=switch(1,$B);

$indexB3=switch(2,$B);

$indexB4=switch(3,$B);

$indexB5=switch(4,$B);

$indexB6=switch(5,$B);
$a1="Anchor shackle";
$a2="Shank";
$a3="Flukes";
$a4="Crown pin";
$a5="Crown plate or head";
$a6="Anchor chain with swivel";
$naam1=switch($indexA1,"$a1","$a2","$a3","$a4","$a5","$a6");
$naam2=switch($indexA2,"$a1","$a2","$a3","$a4","$a5","$a6");
$naam3=switch($indexA3,"$a1","$a2","$a3","$a4","$a5","$a6");
$naam4=switch($indexA4,"$a1","$a2","$a3","$a4","$a5","$a6");
$naam5=switch($indexA5,"$a1","$a2","$a3","$a4","$a5","$a6");
$naam6=switch($indexA6,"$a1","$a2","$a3","$a4","$a5","$a6");
$n1=switch($indexB1,"1","2","3","4","5","6");
$n2=switch($indexB2,"1","2","3","4","5","6");
$n3=switch($indexB3,"1","2","3","4","5","6");
$n4=switch($indexB4,"1","2","3","4","5","6");
$n5=switch($indexB5,"1","2","3","4","5","6");
$n6=switch($indexB6,"1","2","3","4","5","6");
$nummer1=switch($indexA1,"$n1","$n2","$n3","$n4","$n5","$n6");
$nummer2=switch($indexA2,"$n1","$n2","$n3","$n4","$n5","$n6");
$nummer3=switch($indexA3,"$n1","$n2","$n3","$n4","$n5","$n6");
$nummer4=switch($indexA4,"$n1","$n2","$n3","$n4","$n5","$n6");
$nummer5=switch($indexA5,"$n1","$n2","$n3","$n4","$n5","$n6");
$nummer6=switch($indexA6,"$n1","$n2","$n3","$n4","$n5","$n6");
```

Bij elke $naam# hoort een $nummer#.
U kunt in het rolmenu eventueel ook nog meer nummers aanbieden of een greep doen uit een heleboel namen.
In dit schript is veel te varieren.

TIP: Als de lijsten A en B niet gelijk zijn, dan hoeft u geen randomvariabele vooraf te definiëren om A en B niet gelijk te laten zijn.

Zie verder in de handleiding Dynamische plaatjes in de *Handleiding Items maken Deel B* voor het plaatsen van de dynamische nummers of letters.

<img src="....url van het plaatje.....">

$n1

$n2

$n3

$n4

$n5

$n6

TIP: Zie in de *Handleiding Items maken Deel B* hoe u dynamische plaatjes maakt.

1.2.5 Clickable image randomiseren

In de Handleiding *Möbius. Items maken Deel A* staat uitgelegd hoe het vraagtype *Clickable Image* (hotspot) wordt gemaakt.
De student wijst met de muis één van de vooraf gedefinieerde gebieden aan in een plaatje. Hoewel dit soort vragen zwaar op Java leunen en dus ouderwets zijn, worden ze hier toch even aangekaart.
Uitgaande van deze bekendheid, kunnen we nu gaan kijken hoe dit soort vragen te randomiseren zijn.

Figure 1.39: Randomiseren van Clickable Image (eenvoudig)

Het *Algorithm* hiervan is zeer eenvoudig.

$index=rint(3);
$welkerode=switch($index,"meest rechtse","middelste","meest linkse");
$antwoordrood=switch($index,1,2,5);

De vraag is: Zoek van de drie rode ballonnen de $welkerode.

Het antwoord is een getal. Dit getal slaat op het gebied dat als het juiste gebied in de figuur gedefinieerd wordt.
Bekend is dat u bij dit soort vragen een aantal gebieden in het plaatje kunt definiëren met de muis, zoals dat uitgelegd is in de handleiding *Items Maken Deel A* in paragraaf *Vraagtype Clickable Image (hotspot)*.
Als u nu in de broncode van de vraag gaat kijken, dan worden die gebieden (regions) aangegeven met allemaal getallen:

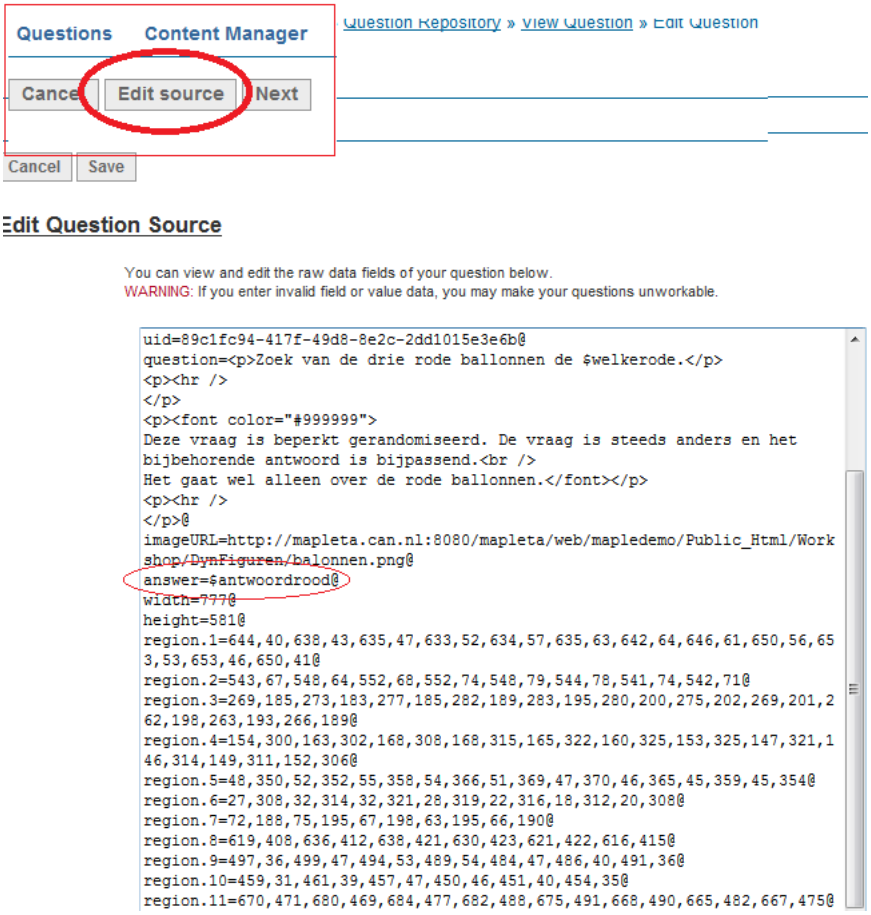


Figure 1.40: Broncode van Clickable Image vraag

Het nummer van de region van het goede antwoord, wordt in de broncode altijd opgenomen in de regel met
answer= # @
Middels uitproberen kunt u te weten komen welke region correspondeert met welke situatie. Handig om te weten is dat de getallen van zo'n region steeds paarsgewijs corresponderen met *x,y*-coördinaten waarbij de *x*-as loopt van boven links naar rechts en de *y*-as loopt van boven links naar beneden! Op die manier hebt u enig idee waar u zich bevindt.
Hier correspondeert "de meest rechtse rode ballon" met region.1 bijvoorbeeld. Het ligt er dus maar aan wat de uiteindelijke vraag is, om te weten wat het bijpassende region is die daarbij hoort. Echter als u dit in de broncode opneemt, zal dat niet lukken, want de vraag moet eerst nog afgemaakt worden.
Maak de vraag dan af nadat u alle regions met de muisklikken hebt vastgesteld en één willekeurig gebied hebt toegewezen als het `goede` antwoord. Probeer nu een en ander uit en achterhaal de nummers van de regions die bij de andere vragen horen zoals de middelste rode ballon en dergelijke. Als u het juiste antwoord wel in het *Algorithm* hebt geprogrammeerd maar eerst nog even een willekeurig gebied als correct definieert en de vraag afsluit, komt daarna de laatste slag.
Exporteer de vraag met het `goede` gedefinieerde antwoord en ga buiten Möbius in een tekstbestand de regel in de broncode aanpassen:

answer=$antwoordrood@

Sla dit tekstbestand op en importeer dit weer in uw *Question Bank*. Altijd wel even controleren natuurlijk, maar de vraag kan daarna niet meer aangepast worden tenzij u hem exporteert, aanpast en weer importeert.

Nog een voorbeeld van hetzelfde plaatje, maar dan met meer vragen en dus ook meer antwoorden:
De vraag luidt: Zoek $ballonnen de $welke ballon en klik deze aan. Het antwoord is een getal: $antwoord.
Het *Algorithm* is het volgende:

```
$indexrood=rint(3);
$antwoordrood=switch($indexrood,1,2,5);
$welkerood=switch($indexrood,"meest rechtse rode","middelste rode","meest linkse rode");
$antwoordgeel=16;
$welkegele="gele";
$indexpaars=rint(3);
$welkepaars=switch($indexpaars,"meest bovenste paarse","middelste paarse","meest onderste paarse");
$antwoordpaars=switch($indexpaars,25,13,24);
$indexantw=rint(3);
$ballonnen=switch($indexantw,"van alle ballonnen","van de drie paarse","van de drie rode");
$welke=switch($indexantw,"$welkegele","$welkepaars","$welkerood");
$antwoord=switch($indexantw,$antwoordgeel,$antwoordpaars,$antwoordrood);
```

Dit is ook weer een voorbeeld van het nesten van de functie switch.

1.3 Randomiseren met berekeningen

Voor bèta's komt hier nog een aantal voorbeelden waarbij er gerandomiseerd wordt bij berekeningen, dus met getallen, grafieken en formules.

1.3.1 Gegeven en gevraagde omwisselen

In het volgende voorbeeld wordt er bijvoorbeeld gevraagd de omrekening te doen van m → cm, maar als deze vraag nogmaals gegenereerd wordt, is het net andersom, zodat de student niet het trucje uit het hoofd kan leren.

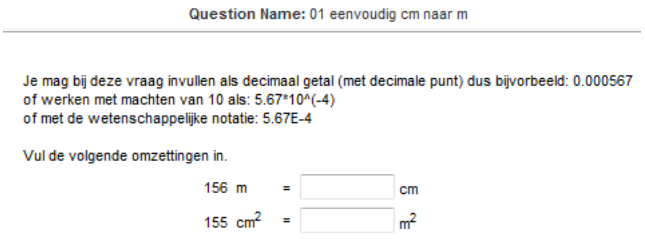


Figure 1.41: Omrekenen, van eenheden vraag en antwoord omwisselen

TIP: In de vraag is er ook bij vermeld op welke manier het antwoord ingevoerd kan worden in het numerieke invulveld. Er zou een link bij moeten staan waarop geklikt kan worden om te zien in welk formaat het antwoord gegeven mag worden, maar daar moet MapleSoft nog in voorzien en is in versie 9 al wel gerealiseerd.
Het bijbehorende *Algorithm* is als volgt gebouwd:

```
$cm_m=10^(-2);
$m_cm=10^2;
$cm2_m2=10^(-4);
$m2_cm2=10^4;
$a=range(3,300);
$b=range(3,300);
$c=range(3,300);
$d=range(3,300);
$index1=rint(2);
$cmma=switch($index1,"cm","m");
$cmmb=switch($index1,"m","cm");
$eenheidcmm=switch($index1,$cm_m,$m_cm);
```

```
$index2=rint(2);
$cmm2a=switch($index2,"cm<sup>2</sup>","m<sup>2</sup>");
$cmm2b=switch($index2,"m<sup>2</sup>","cm<sup>2</sup>");
$eenheidcmm2=switch($index2,$cm2_m2,$m2_cm2);
```

Er zijn twee numerieke invulvelden. Voor het eerste invulveld geldt $index1 waar gewisseld wordt tussen cm en m met bijbehorende omrekenfactor. En voor het tweede invulveld geldt $index2.

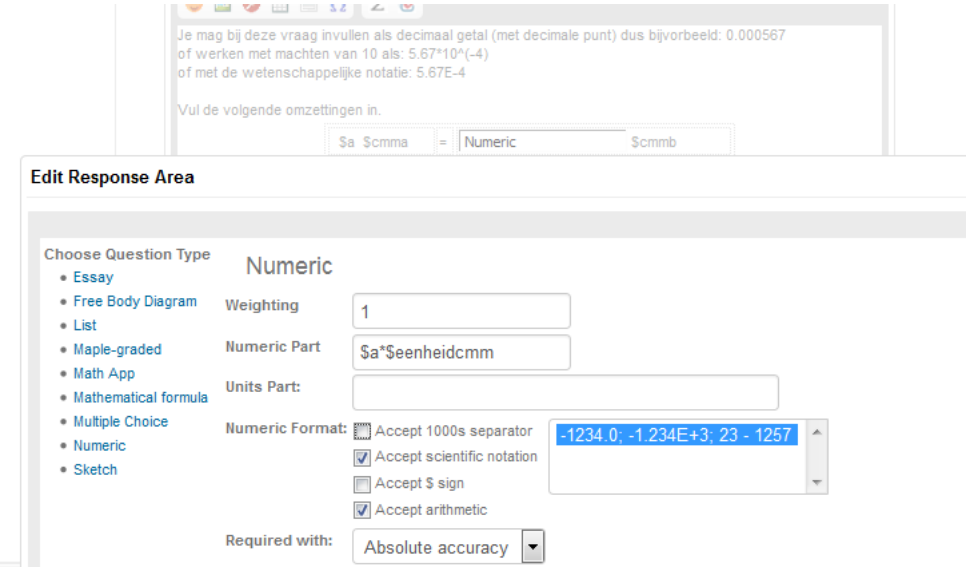


Figure 1.42: Numeriek invulveld met antwoord als berekening

TIP: Let hier op de instellingen van het Numerieke antwoordveld. In de bovenstaande figuur zien we de instellingen zonder duizendtalseparator en mét wetenschappelijke notatie en ook met *Accept arithmetic* en dat bij het *Numeric Part* het antwoord als berekening is gegeven. Dat laatste heeft ermee te maken dat als de student de vraag fout beantwoord heeft, dat dan ook de berekening van het correcte antwoord wordt weergegeven.
We zien dat hier steeds de eenheden omwisselen van gegeven naar gevraagd.

Nog een voorbeeld waarbij steeds de gegevens en de vraag omgewisseld worden.

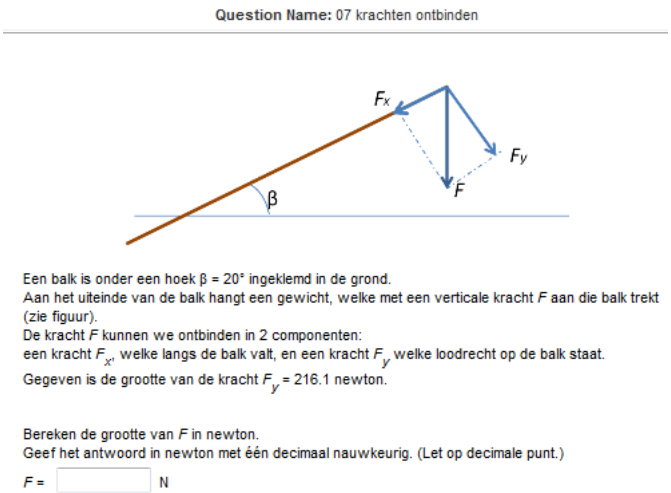


Figure 1.43: Gegeven en gevraagde omwisselen

Het *Algorithm* dat hierbij hoort is

```
$F=range(200,400,10);
$beta=range(20,30);
$Fx=maple("evalf[4]($F*sin($beta*Pi/180))");
$Fy=maple("evalf[4]($F*cos($beta*Pi/180))");
$index1=range(0,2);
$gegeven=switch($index1,$F, $Fx,$Fy);
$gegevenkracht=switch($index1,"F","F<sub>x</sub>","F<sub>y</sub>");
$bereken=switch($index1,"F<sub>x</sub>", "F<sub>y</sub>", "F");
$antwoord=switch($index1,"$Fx", "$Fy", "$F");
```

Hierin laten we Maple alle componenten berekenen in 4 significante cijfers.
Maple rekent met hoeken in radialen, dus de graden van de hoek β moeten nog omgerekend worden naar radialen. Bij Maple is Pi standaard de π die bedoeld wordt.
Met $index1 laten we steeds alles in de pas lopen en wisselen gegeven en gevraagde steeds met elkaar.
Let wel op afrondingen, dus geef een ruime marge voor het antwoord.
TIP: Let hier op de html-tags die al in het *Algorithm* kunnen worden meegegeven. In de tekst van de vraag kan met cursief gewerkt worden.

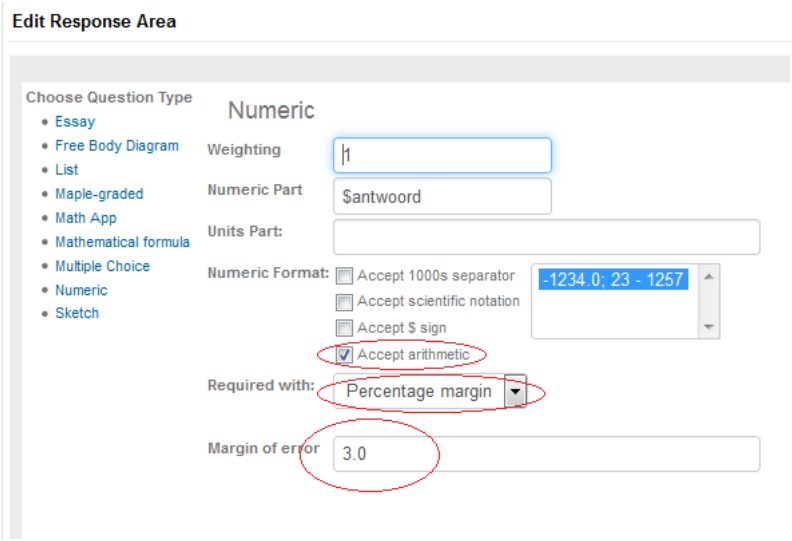


Figure 1.44: Numeriek invulveld met tolerantie percentage

1.3.2 Getallen

Random getallen kunnen gemakkelijk in het *Algorithm* gegenereerd worden.
Het *Algorithm* is heel eenvoudig en als u de codes nog niet goed kent, is er altijd nog de *Designer* van het *Algorithm* om de variabelen met behulp van een wizard te maken. Zie voor deze *Designer* in paragraaf *Gebruik van de Designer in het Algorithm (page 52)*.

```
$a=range(200,800);
$b=decimal(3,rand(0.1,0.999));
$ab=$a*$b;
```

TIP: Let ook op dat u bij getallen boven de 1000 ervoor waakt dat er in de presentatie van de vraag en eventuele feedback of hints geen duizendtalseparator verschijnt. Soms wilt u ook voorkomen dat breuken omgezet worden in decimalen en dat kan allemaal met behulp van quotes zoals het volgende laat zien:

```
$a=range(200,800);
$b=decimal(3,rand(0.1,0.999));
$ab=$a*$b;
$c=3/4;
$c1="3/4";
$d=range(2000,3000);
$d1="$d";
```

Variable	Value
a	277
b	0.639
ab	177.003
AB	177.003
c	0.75
c1	3/4
d	2,107
d1	2107

Figure 1.45: Getallen in het Algorithm

In het volgende voorbeeld moeten er twee breuken met ongelijke noemers worden opgeteld. Met het *Maple-graded* vraagtype wordt het antwoord overhoord waarbij de student het juiste vereenvoudigde antwoord moet invullen in een editor waar de Preview-knop niet aanwezig is.
TIP: Let op dat u hier de *Equation Editor* aanbiedt bij de formule-instellingen van deze vraag, want als u dat niet doet, dan kan de student na het intikken op de Preview-knop klikken waarbij het vereenvoudigde antwoord getoond wordt en dat is hier dus niet de bedoeling.

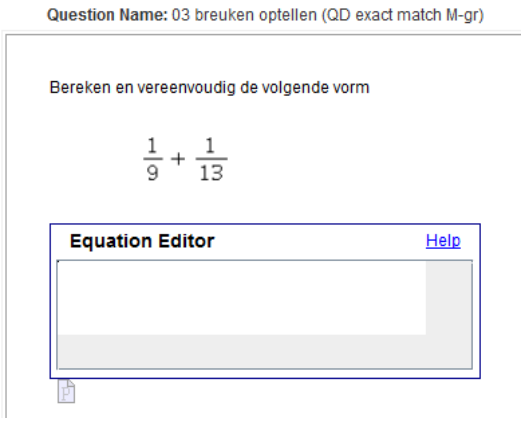


Figure 1.46: Breuken optellen

Het bijbehorende *Algorithm* is

```
$a1=range(2,15);
$b=range(2,20);
$a=if(ne(($a1),($b)),($a1),($a1)+1);
$vraag=mathml("1/($a)+1/($b)");
$antwoord=maple("simplify(1/($a)+1/($b))");
$stringantwoord=maple("convert($antwoord,string)");
$antwoorddisplay=maple("printf(MathML:-ExportPresentation(($antwoord)))");
```

In deze code wordt ervoor gezorgd dat de noemers van de twee breuken in ieder geval niet dezelfde zijn.

De $vraag is voor deze eenvoudige formule MathML gecodeerd met behulp van mathml("....").
Het $antwoord is de vereenvoudiging van deze optelling. Ook wordt het antwoord naar een string omgezet, want bij de grading willen we het antwoord van de student daarmee matchen, zodat de student het meest vereenvoudigde antwoord moet geven. Een student wil nog wel eens een spatie tikken links en rechts van een operator.

Grading Code:

with(StringTools); stringrespons := Remove(IsSpace, "$RESPONSE"):
evalb(SubString(stringrespons, 1..-1)=$stringantwoord);

Nadat eventuele door de student ingetikte spaties verwijderd zijn, moet het antwoord van de student van het eerste tot en met het laatste karakter precies overeenkomen.
Bij het correcte antwoord geven we de 2D presentatie van het antwoord.

Answer: printf(MathML:-ExportPresentation(($antwoord)));

1.3.3 Feedback afhankelijk van de situatie

In het volgende voorbeeld bereiden we feedback voor die gebruikt kan worden na afloop van de grading. Deze wordt ingevuld in de rubriek *Feedback* en komt als *Comment* na afloop van de grading zichtbaar voor de student.
De student hoeft alleen maar een getal in te vullen in het numerieke invulveld.

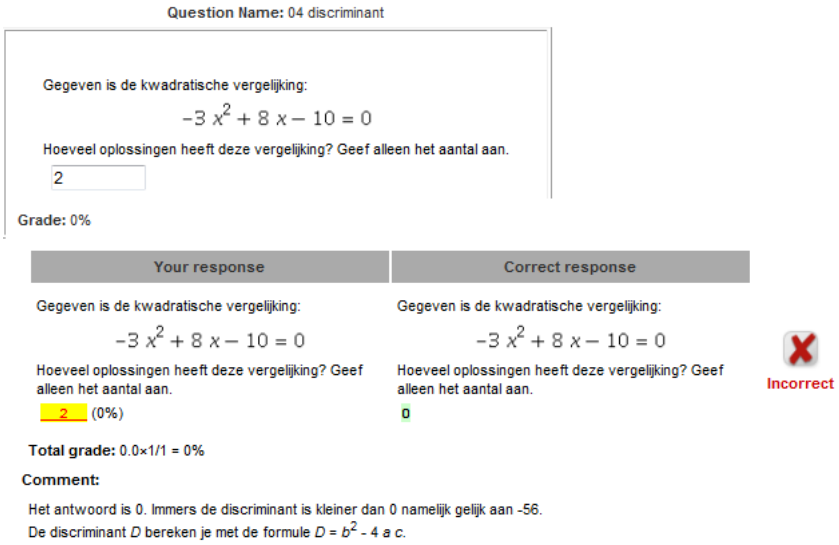


Figure 1.47: Feedback afhankelijk van de situatie

Het bijbehorende *Algorithm* is:

$a=switch(rint(2),range(-10,-1),range(1,10));
$b=switch(rint(2),range(-10,-1),range(1,10));
$c=switch(rint(2),range(-10,-1),range(1,10));
$verg=maple("($a)*x^2+($b)*x+($c)=0");
$vergdisplay=maple("printf(MathML[ExportPresentation]($verg))");
$GGD=maple("igcd($a,$b,$c)");
condition:eq($GGD,1);
$discr=maple("discrim(lhs($verg),x)");
$antw=maple("if ($discr)<0 then 0 elif ($discr)=0 then 1 elif ($discr)>0 then 2 end if");
$uitleg=switch($antw,"de discriminant is kleiner dan 0","de discriminant is gelijk aan 0","de discriminant is groter dan 0");

Er wordt voor gezorgd dat de coëfficiënten niet gelijk zijn aan 0 en verder ook geen gemeenschappelijke deler hebben.
(igcd = greatest common divisor of integers)

In de *Feedback* communiceren we het volgende:

Het antwoord is $antw. Immers $uitleg namelijk gelijk aan $discr.

De discriminant D bereken je met de formule $D = b^2 - 4\,a\,c$.
zodat steeds de feedback zich aanpast aan de situatie.

1.3.4 Matching met formules en dynamische grafieken

In de volgende serie voorbeelden moeten grafieken aan functies gekoppeld worden. Er zijn vier functievoorschriften en de grafieken bereiden we ook in het *Algorithm* voor.
Het vraagtype *Matching* werkt hier het snelst maar geeft alleen de mogelijkheid om evenveel gegevens aan evenveel antwoorden te koppelen. Dat gaat met het invullen van corresponderende nummertjes zoals in onderstaande figuur te zien is.
Zie voor het vraagtype *Matching* ook in paragraaf *Vraagtype Matching met tekstrandomisering* *(page 16)*.

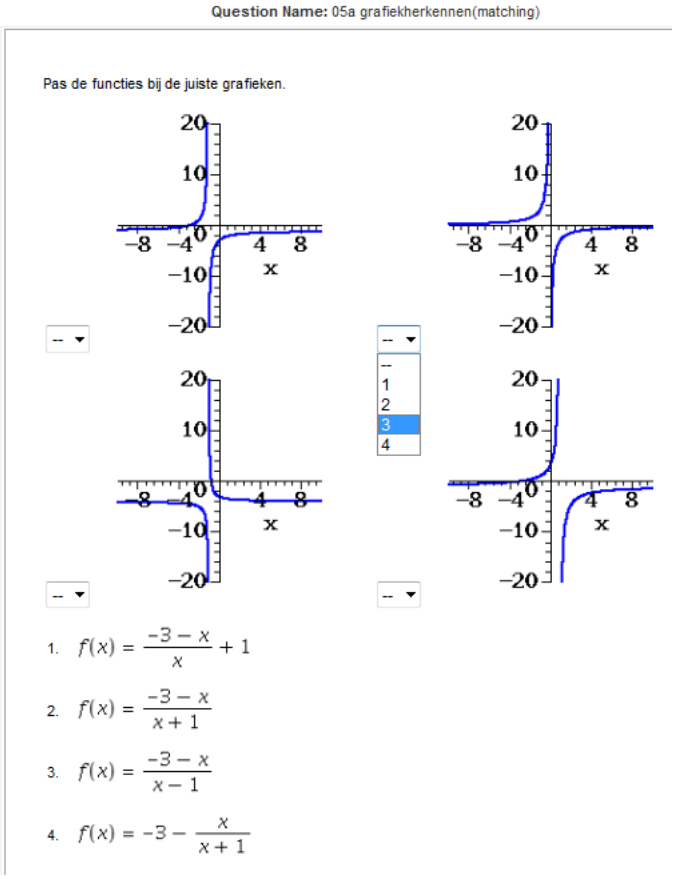


Figure 1.48: Matching met dynamische figuren en formules

Het *Algorithm* kan vrij eenvoudig gehouden worden omdat in het vraagtype zelf de randomisering al automatisch geregeld wordt. Echter dit vraagtype *Matching* is wat beperkt in vergelijking met de *Question Designer* waar meer mogelijkheden aanwezig zijn, dus laten we verderop ook zien hoe dit type vragen in de *Question Designer* gemaakt kunnen worden. Het *Algorithm* is als volgt:

```
$a=switch(rint(2),range(-4,-1),range(1,4));
$b=switch(rint(2),range(-4,-1),range(2,4));
$c=range(1,4);
condition:not(eq($a,$b));
$kleur=switch(rint(6),"red","green","blue","magenta","navy","aquamarine");
$f=maple("$a+($b)*x/(x+($c))");
$displayf=maple("printf(MathML[ExportPresentation](f(x)=$f))");
$g=maple("($a+($b)*x)/(x+($c))");
```

```
$displayg=maple("printf(MathML[ExportPresentation](f(x)=$g))");
$h=maple("($a+($b)*x)/x+($c)");
$displayh=maple("printf(MathML[ExportPresentation](f(x)=$h))");
$k=maple("($a+($b)*x)/(x-($c))");
$displayk=maple("printf(MathML[ExportPresentation](f(x)=$k))");
$grafiekf=plotmaple("plot($f, x=-10..10,-20..20,thickness=2,discont=true,color=$kleur), plotoptions='height=200, width=200' ");
$grafiekg=plotmaple("plot($g, x=-10..10,-20..20,thickness=2,discont=true,color=$kleur), plotoptions='height=200, width=200' ");
$grafiekh=plotmaple("plot($h, x=-10..10,-20..20,thickness=2,discont=true,color=$kleur), plotoptions='height=200, width=200' ");
$grafiekk=plotmaple("plot($k, x=-10..10,-20..20,thickness=2,discont=true,color=$kleur), plotoptions='height=200, width=200' ");
```

Er worden in het *Algorithm* vier functies aangemaakt die op elkaar lijken maar steeds wat andere parameters hebben.
Vervolgens worden er MathML-coderingen gemaakt van de formules ten behoeve van de formules in de vraag.
De grafieken worden met Maple gemaakt. Als u een beetje kennis heeft van het maken van grafieken in Maple, laat u Maple dus het werk doen. Let ook eens op de variabele $kleur. Met een kleine moeite zijn de grafieken ook steeds anders van kleur. In de *Designer* van het *Algorithm* kunt u de code opvragen voor het maken van zo'n grafiek: plotmaple("....."). Zie ook paragraaf *Gebruik van de Designer* *(page 52)* voor het maken van een grafiek.

Als u de *Question Designer* wilt gebruiken in plaats van het vraagtype *Matching*, dan zal het gebruik van drop-down menu's niet gaan, want het vraagtype *List* kan alleen maar getallen en tekst aan en geen formules en geen plaatjes en ook geen html-codes. Het makkelijkst is dan om *Multiple Choice* aan te bieden.
U bent nu in feite niet meer gebonden aan slechts evenveel gegevens als antwoorden. U kunt nu bij ieder gegeven uit meer antwoorden laten kiezen wat in feite de gokkans naar beneden haalt.

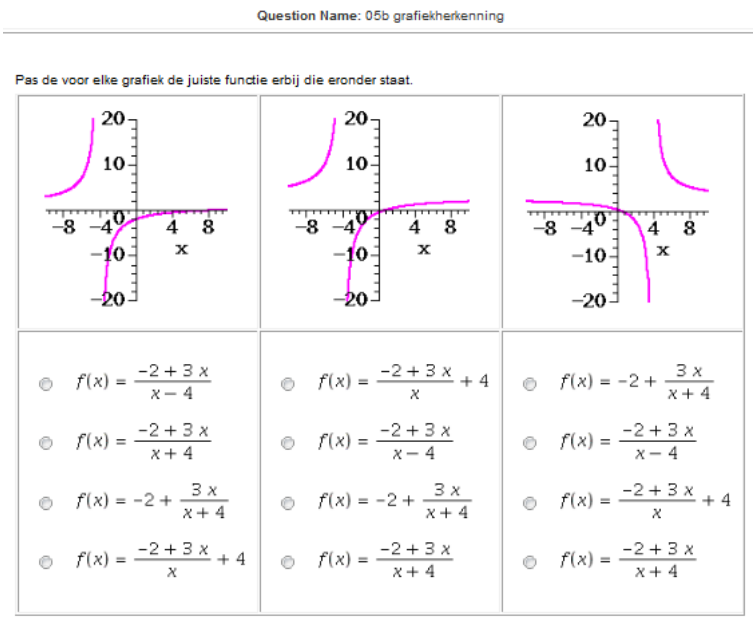


Figure 1.49: Dynamische grafieken en formules in matchingssituatie in de Question Designer

Het bijbehorende *Algorithm* is nu iets ingewikkelder omdat nu geen automatische randomisering plaats vindt zoals in het vraagtype *Matching*. Formules en plaatjes moeten nu van te voren in het *Algorithm* reeds doorelkaar gehusseld worden.

```
$a=switch(rint(2),range(-4,-1),range(1,4));
$b=switch(rint(2),range(-4,-1),range(2,4));
$c=range(1,4);
condition:not(eq($a,$b));
$kleur=switch(rint(6),"red","green","blue","magenta","navy","aquamarine");
$f=maple("$a+($b)*x/(x+($c))");
$g=maple("($a+($b)*x)/(x+($c))");
```

```
$h=maple("($a+($b)*x)/x+($c)");
$k=maple("($a+($b)*x)/(x-($c))");
$lijstA=maple("[0,1,2,3]");
$A=maple("randomize():StringTools[Randomize]():combinat[randperm]($lijstA)");

$index1=switch(0,$A);

$index2=switch(1,$A);

$index3=switch(2,$A);

$index4=switch(3,$A);
$F1=switch($index1,"$f","$g","$h","$k");
$F2=switch($index2,"$f","$g","$h","$k");
$F3=switch($index3,"$f","$g","$h","$k");
$F4=switch($index4,"$f","$g","$h","$k");
$displayF1=maple("printf(MathML[ExportPresentation](f(x)=$F1))");
$displayF2=maple("printf(MathML[ExportPresentation](f(x)=$F2))");
$displayF3=maple("printf(MathML[ExportPresentation](f(x)=$F3))");
$displayF4=maple("printf(MathML[ExportPresentation](f(x)=$F4))");
$grafiekF1=plotmaple(" plot($F1, x=-10..10,-20..20,thickness=2,discont=true,color=$kleur), plotoptions='height=200, width=200' ");
$grafiekF2=plotmaple(" plot($F2,x=-10..10,-20..20,thickness=2,discont=true,color=$kleur), plotoptions='height=200, width=200' ");
$grafiekF3=plotmaple(" plot($F3, x=-10..10,-20..20,thickness=2,discont=true,color=$kleur), plotoptions='height=200, width=200' ");
$grafiekF4=plotmaple(" plot($F4, x=-10..10,-20..20,thickness=2,discont=true,color=$kleur), plotoptions='height=200, width=200' ");
```

Het komt er dus op neer dat eerst de functies door elkaar gegooid worden en daarna worden er formules en plaatjes van gemaakt.
TIP: Let ook eens op de tabel in de tekst van de vraag met breedte 600 pixels vanwege het feit dat elke grafiek 200 pixels breed is.

In de editor van de vraag ziet het er als volgt uit:

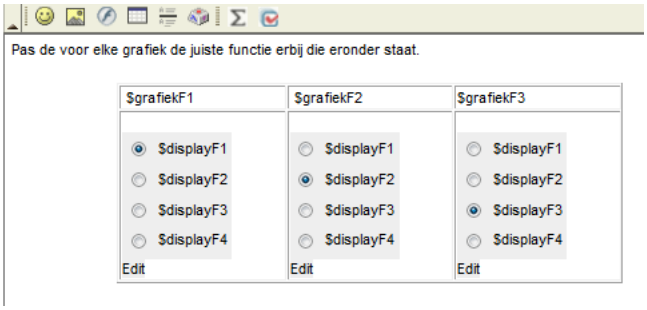


Figure 1.50: Grafieken in een tabel met daaronder MC vragen

1.3.5 Multiple choice met berekeningen

Bij berekeningen is het handig om altijd even wat meer goede en foute antwoorden te programmeren zodat de student op den duur niet het idee kan opperen om bijvoorbeeld altijd bij dit soort vraag het op één na grootste antwoord te kiezen.

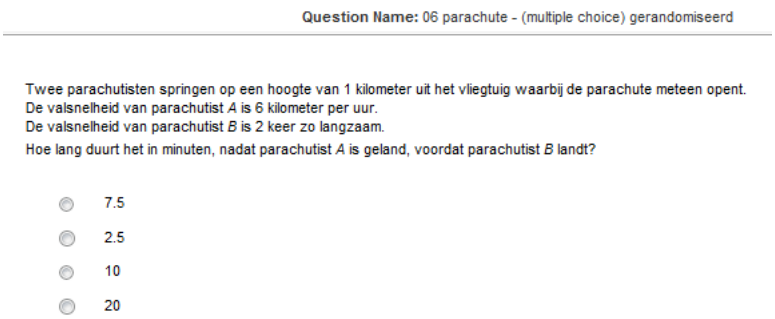


Figure 1.51: Multiple Choice met berekeningen

Het bijbehorende *Algorithm* is als volgt:

```
$h=1;
$vA=switch(rint(4),5,6,10,12);
$k=switch(rint(3),4,3,2);
$vB=maple("$vA/$k");
$tA=maple("$h/($vA)");
$tB=maple("$h/($vB)");
$verschilminuten=maple("($tB-$tA)*60");
$antwf1=$verschilminuten/2;
$antwf2=3*$verschilminuten/2;
$antwf3=3*$verschilminuten/4;
$antwf4=2*$verschilminuten;
$antwf5=$verschilminuten/4;
$antwf6=3*$verschilminuten;
$A=maple("randomize():combinat[randcomb]([0,1,2,3,4,5],3)");

$index1=switch(0,$A);

$index2=switch(1,$A);

$index3=switch(2,$A);
$Antwf1=switch($index1,"$antwf1","$antwf2","$antwf3","$antwf4","$antwf5","$antwf6");
$Antwf2=switch($index2,"$antwf1","$antwf2","$antwf3","$antwf4","$antwf5","$antwf6");
$Antwf3=switch($index3,"$antwf1","$antwf2","$antwf3","$antwf4","$antwf5","$antwf6");
```

Hier is slechts één antwoord goed ($verschilminuten) en voor de rest zijn er hier 6 foute antwoorden geprogrammeerd. Daaruit wordt iedere keer een greep van 3 genomen. De rest wijst zich vanzelf. Dit soort dingen kunt u vaak gebruiken!
Let ook eens op het gebruik van switch om een keuze te maken uit een set speciale getallen in plaats van het opgeven van een range.

Het volgende voorbeeld is ook van het type *Multiple Choice* maar iets ingewikkelder.

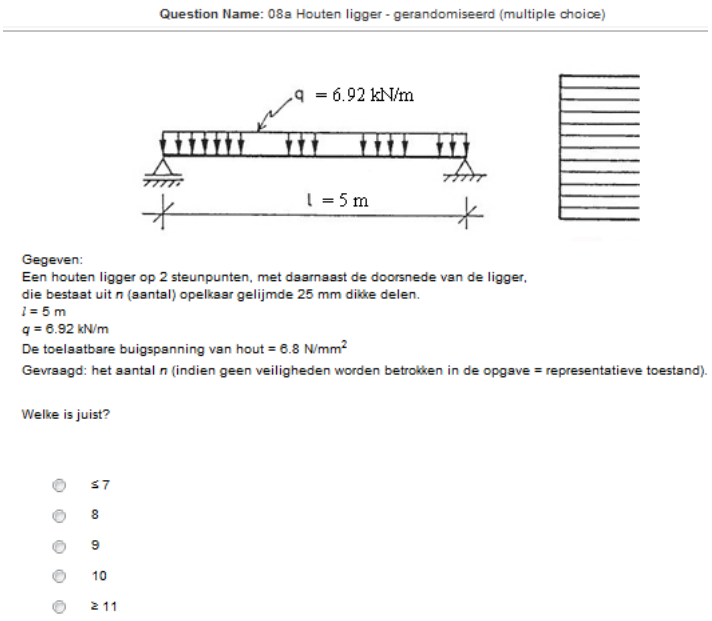


Figure 1.52: Multiple Choice met variabele grading

Deze vraag is van het type *Multiple Choice* binnen de *Question Designer*. Er is echter geen vaste keuze voor het antwoord. Het is dus bijvoorbeeld ook niet zo dat bij deze vraag altijd het één na grootste antwoord gekozen moet worden of iets dergelijks. In feite

"beweegt" het goede antwoord over de 5 aangeboden antwoorden heen en weer elke keer als de vraag wordt geopend. Wel blijven de afleiders op volgorde staan hoewel ze steeds anders van grootte zijn.
Deze vraag is verder gerandomiseerd wat betreft de gegevens van de lengte van de balk en de lineaire belasting. De toelaatbare spanning van hout varieert ook iets. De gegevens in het plaatje veranderen mee. Daarvan is gebruikgemaakt van *Labeled Image* waarover in de *Handleiding Items maken Deel B* gesproken wordt bij het hoofdstuk van *Dynamisch figuren*.
(In de feedback van deze vraag is de volledige oplossing met formules gegeven waarbij de waarden van de variabelen aangepast zijn.)
Het bijbehorend *Algorithm* is ontdaan van variabelen ten behoeve van de feedback en het uiteindelijke goede antwoord is $N.
We willen in feite 5 afleiders hebben voor de *Multiple Choice*-vraag die we op volgorde aan willen bieden.

```
$b=range(160,200,10);
$d=range(20,25);
$h=maple("n*$d");
$q=decimal(2,rand(6.8,7.3));
$L=range(3,7);
$sigmatoel=decimal(1,rand(6.5,8.5));
$I=maple("1/12*$b*($h)^3");
$Mmax=maple("1/8*$q*($L*10^3)^2");
$sigmastrip=maple("$Mmax*($h/2)/($I)*n^2");
$N=maple("ceil(sqrt($sigmastrip/$sigmatoel)-0.05)");
$index=rint(9);
$antw1=switch($index, $N-6,$N-5,$N-4,$N-3,$N-2,$N-1,$N,$N+1,$N+2);
$antw2=switch($index+1,$N-6,$N-5,$N-4,$N-3,$N-2,$N-1,$N,$N+1,$N+2,$N+3);
$antw3=switch($index+2,$N-6,$N-5,$N-4,$N-3,$N-2,$N-1,$N,$N+1,$N+2,$N+3,$N+4,$N+5);
$antw4=switch($index+3,$N-6,$N-5,$N-4,$N-3,$N-2,$N-1,$N,$N+1,$N+2,$N+3,$N+4,$N+5,$N+6);
$antw5=switch($index+4,$N-6,$N-5,$N-4,$N-3,$N-2,$N-1,$N,$N+1,$N+2,$N+3,$N+4,$N+5,$N+6,$N+7);
$Antw=switch($index,5,5,5,4,3,2,1,1,1,1);
```

Stel voor dat $index gelijk is aan 4, dan is $antw1 het juiste antwoord $N min 2. Dan is $antw2 het juiste antwoord min 1 en $antw3 is het juiste antwoord. Verder is $antw4 het juiste antwoord plus 1 en $antw5 is het juiste antwoord plus 2. Kortom het is dus het derde antwoord.
Stel dat $index gelijk is aan 8, dan is $antw1 het juiste antwoord +2 en $antw2 het juiste antwoord +3 enzovoort. Dus dan is het eerste antwoord dus goed. Immers $N < $N+2.

TIP: Let nu wel op dat de instellingen van het *Multiple Choice*-invulveld nu op *Non Permuting* staan, want het is hier wel mooi om de juiste volgorde van de getallen aan te bieden.

1.4 Strings manipuleren

Een mooi voorbeeld van het gebruik van het Maple-pakket StringTools is het volgende:
In het *Algorithm* worden hier verschillende variabelen van bijvoorbeeld een DNA-streng gegenereerd.
Het is mogelijk om een willekeurig aantal letters achter elkaar te plaatsen en vervolgens hier veranderingen in aan te brengen.

```
$testDNA=maple("with(StringTools):Randomize():Random(13,convert(acgt,string))");
$testDNA0=$testDNA;
$rotatie=maple("StringTools[Rotate]("$testDNA",3)");
$achterstevoren=maple("StringTools[Reverse]("$testDNA")");
$subs=maple("with(StringTools):
s1:=SubstituteAll("$testDNA",convert(t,string),convert(1,string)):
s2:=SubstituteAll(s1,convert(a,string),convert(t,string)):
s3:=SubstituteAll(s2,convert(1,string),convert(a,string)):
s4:=SubstituteAll(s3,convert(g,string),convert(1,string)):
s5:=SubstituteAll(s4,convert(c,string),convert(g,string)):
SubstituteAll(s5,convert(1,string),convert(c,string))");
$subs0=$subs;
```

Hieronder is het effect ervan te zien:

```
$testDNA=maple("with(StringTools):Randomize():Random(13,convert
(acgt,string))");
$testDNA0=$testDNA;
$rotatie=maple("StringTools[Rotate]("$testDNA",3)");
$achterstevoren=maple("StringTools[Reverse]("$testDNA")");
$subs=maple("with(StringTools):
s1:=SubstituteAll("$testDNA",convert(t,string),convert(1,string)):
s2:=SubstituteAll(s1,convert(a,string),convert(t,string)):
s3:=SubstituteAll(s2,convert(1,string),convert(a,string)):
s4:=SubstituteAll(s3,convert(g,string),convert(1,string)):
s5:=SubstituteAll(s4,convert(c,string),convert(g,string)):
SubstituteAll(s5,convert(1,string),convert(c,string))");
$subs0=$subs;
```

Variable	Value
testDNA	"atgcacttccttg"
testDNA0	atgcacttccttg
rotatie	"ttgatgcacttcc"
achterstevoren	"gttccttcacgta"
subs	"tacgtgaaggaac"
subs0	tacgtgaaggaac

Figure 1.53: Manipuleren van strings

In de eerste regel wordt hier een willekeurige string gemaakt van de letters *a*, *c*, *g* en *t* ter totale lengte van 13.
Het is niet te sturen hoeveel malen elk karakter hierin voorkomt, maar als u dat onwenselijk vindt, is er nog wel een andere mogelijkheid. (Dat wordt hieronder uitgelegd.)
Met deze willekeurige string (combinatie van karakters tussen dubbele quotes), kan Maple een aantal dingen voor ons doen door middel van comando's uit het StringTools-pakket. Als we echter de karaktercombinatie op het scherm nodig hebben zonder quotes, dan is dat eenvoudig te realiseren door een nieuwe variabele $testDNA0=$testDNA te definiëren. De quotes worden dan gestript.
Echter om Maple iets met de string te laten doen, moet deze wel van het formaat string zijn, dus mét quotes.
Een rotatie kan verwezenlijkt worden met het commando *StringTools[Rotate]* waarbij een cyclische verwisseling ontstaat, in dit geval worden alle karakters 3 naar rechts verplaatst.
De string kan eenvoudig achterstevoren gezet worden met *StringTools[Reverse]*.
Het is ook mogelijk om substitutie toe te passen metSubstituteAll uit het StringTools pakket. (Dus eerst wel het StringTools-pakket aanroepen.
In *SubstituteAll*("string","a","b") worden alle letters *a* in de string vervangen door de letter *b*.
TIP: Past u de substitutie toe op een reeds gedefinieerde string, dan kunt u die string aangeroepen en niet vergeten er quotes omheen te plaatsen. Echter voor de string "a" vult u in convert(a,string), en ook op dezelfde manier voor "b" want anders begrijpt het systeem al die quotes niet.
Als er meer substituties moeten plaatsvinden dan kan dat alleen door achtereenvolgens een aantal substituties achter elkaar uit te voeren. Hier is dus achtereenvolgens de *a* veranderd in een *t* en andersom en de *c* is veranderd naar een *g* en andersom door een aantal substituties achter elkaar. Er zit niets anders op dan dat in 6 stappen te doen.

In het volgende voorbeeld ziet u hoe de string gepermuteerd kan worden volgens een vooraf vastgestelde lijst.

```
$A=maple("StringTools[Randomize]():combinat[randperm]
([1,2,3,4,5,6,7,8,9,10,11,12,13])");
$permutatie=maple("StringTools[Permute](convert($testDNA0,string),$A)");
$teststring=maple("convert(aabbccdd,string)");
$teststring0=$teststring;
$AA=maple("combinat[randperm](length("$teststring"))");
$testperm=maple("StringTools[Permute](convert($teststring0,string),$AA)");
```

A	[10, 4, 7, 9, 6, 12, 11, 8, 3, 2, 13, 1, 5]
permutatie	"gggattgcggggg"
teststring	"aabbccdd"
teststring0	aabbccdd
AA	[8, 6, 3, 4, 2, 7, 1, 5]
testperm	"dcbbadac"

Figure 1.54: Permutaties van een string

De string kan gepermuteerd worden waarbij dezelfde karakters als in de originele string omgewisseld worden volgens een lijst die aangeeft op welke plaats elk karakter komt te staan.
Deze lijst moet natuurlijk bestaan uit evenveel elementen als het aantal karakters in de string. Op deze manier kan de permutatie gestuurd worden.
Wilt u echter een willekeurige permutatie, dan kan de lijst vooraf gerandomiseerd worden en door het aanroepen van de gerandomiseerde lijst bij *StringTools*[*Permute*], hebt u een willekeurige permutatie van de string.
In bovenstaande figuur ziet u ook hoe te beginnen met een string van karakters waarvan u zelf de aantallen van de verschillende karakters wilt bepalen. Ook is het handig als u deze string wilt permuteren, dat u dan een passende lijst maakt die meteen gegenereerd wordt naar aanleiding van het aantal karakters in betreffende string. Met het gewone Maple-commando *length* kunt u de lengte van de string bepalen (*length* werkt buiten elk pakket om) en met het commando *combinat[randperm](#)* krijgt u standaard een lijst met de getallen 1 tot en met # tevoorschijn.
TIP: Roep bij het commando *Permute* uit het StringTools-pakket de string niet direct aan maar converteer eerst de karaktercombinatie zonder quotes naar een string binnen het Permute-commando.

TIP: Kijk ook eens in de *Handleiding Items Maken Deel B* waar gebruikgemaakt wordt van strings bij de *Grading Code* van het Maple-graded vraagtype.

1.4.1 Lijstje met StringTool commando's

Buiten elk pakket kunt u sommige dingen met strings doen:

- **length**(string) telt het aantal karakters van een string.
- **cat**("string1","string2","string3") koppelt de strings aan elkaar. Echter het hoeven geen strings te zijn, het kunnen ook namen zijn.

Binnen het pakket StringTools kunnen de volgende commando's handig zijn.
with(stringtools):

- **CaseJoin**([string1,string2,string3]) de elementen van een lijstje met strings worden aan elkaar vastgeknoopt tot een string, waarbij het begin steeds een hoofdletter is.
- **CountCharacterOccurrences**("a+p*b+pq+q^2-1/q+1/pq","q") telt het aantal keren dat het enkele karakter q in de string voorkomt. Dit kan alleen met enkele karakters. Moeten er karaktercombinaties geteld worden, gebruik dan SearchAll (zie onder).
- **Explode**("abcdefg") geeft een lijst met allemaal strings van de afzonderlijke karakters en werkt dus omgekeerd aan Implode.
- **Implode**(["a","b","c"]) geeft "abc". Dit commando werkt alleen op een lijstje met strings die bestaan uit één karakter. Er wordt een string gevormd van deze karakters. Als de lijst strings bevat van meer karakters wordt steeds alleen het eerste karakter meegenomen. Implode werkt tegenovergesteld aan Explode.
- **Length**(string) geeft het aantal karakters van een string.
- **Join**([string1,string2,string3],"") de elementen van een lijstje met strings worden aan elkaar vastgeknoopt tot een string. Als er niet de optie "" gegeven wordt, dan komt er een spatie tussen de strings te staan.

- **Permute**(string,lijst) gooit de volgorde van de karakters van de string door elkaar volgens de nummers van de lijst. Let op dat de lijst even lang is als het aantal karakters van de string.
- Randomize(): **Random**(2,string) neemt twee willekeurige karakters van de string.
- **Remove**(karakter, string) U kunt één of meer karakters uit een string verwijderen. Deze karakters gelden dan niet als combinaties. Dus als u "cd" uit een string verwijdert, dan worden alle c's en alle d's verwijderd uit de string. Ook een spatie geldt als karakter en die kunt u dus ook verwijderen.
- **Reverse**(string) karakters achterstevoren zetten.
- **Rotate**(string,3) cyclische verwisseling, alles gaat 3 *vooruit*.
- **SubstituteAll**(string,"t","1") vervangt alle t in de string door een 1.
  Er kunnen eventueel ook karaktercombinaties vervangen worden door een andere combinatie.
- **Search**("/","a+p*b+pq+q^2-1/q+1/pq") zoekt het karakter in een string en geeft aan op welke plaats het voor de eerste keer optreedt. Als het karakter niet gevonden wordt, dan wordt 0 weergegeven.
- **SearchAll**("pq","a+p*b+pq+q^2-1/q+1/pq") geeft een rijtje op welke plaats(en) de karaktercombinatie pq optreedt.
- **nops**({StringTools[**SearchAll**]("pq","a+p*b+pq+q^2-1/q+1/pq")}) telt het aantal malen dat de karaktercombinatie pq in de string optreedt.

```
> length("dfgryuiopppp");
```

12

```
> combinat[randperm](12);
```

[8, 6, 7, 3, 10, 2, 5, 11, 4, 1, 9, 12]

```
> StringTools[SubstituteAll]("abctred","t","1");
```

"abc1red"

```
> StringTools[SubstituteAll]("abctred","tr","1000");
```

"abc1000ed"

```
> StringTools[Join](["string1","string2","string3"]);
```

"string1 string2 string3"

```
> StringTools[Join](["string1","string2","string3"],"");
```

"string1string2string3"

```
> StringTools[CaseJoin](["string1","string2","string3"]);
```

"String1String2String3"

```
> cat("string1","string2","string3");
```

"string1string2string3"

```
> cat(ab,cde);
```

abcde

```
> StringTools[Rotate]("abcde",3);
```

"cdeab"

```
> StringTools[Implode](["ac","b","v"]);
```

"abv"

```
> StringTools[Explode]("abcdefg");
```

["a", "b", "c", "d", "e", "f", "g"]

```
> StringTools[Remove](" ","ab cde fg");
```

"abcdefg"

```
> StringTools[Remove]("cd","ab cde fg ck d");
```

"ab e fg k "

```
> StringTools[SearchAll]("pq","a+p*b+pq+q^2-1/q+1/pq");
```

7, 20

```
> nops({StringTools[SearchAll]("pq","a+p*b+pq+q^2-1/q+1/pq")});
```

2

```
> StringTools[Search]("/","a+p*b+pq+q^2-1/q+1/pq");
```

15

```
> StringTools[CountCharacterOccurrences]("a+p*b+pq+q^2-1/q+1/pq","q");
```

4

```
> StringTools[Search]("p","a+p*b+pq+q^2-1/q+1/pq");
```

3

1.5 Lijsten manipuleren

Een aardig voorbeeld om achtereenvolgens dingen op volgorde te laten zetten (plaatjes, formules, of andere zaken):

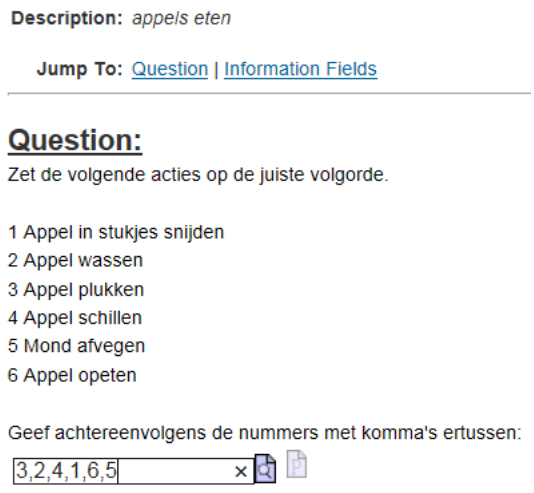


Figure 1.55: Op volgorde zetten

Deze vraag is gerandomiseerd, dat wil zeggen dat de zes acties steeds willekeurig onder elkaar worden gezet:
Het is een Maple-graded vraag en met Maple kan de sequentie van de volgorde gecheckt worden.

In het *Algorithm* is het volgende geprogrammeerd:

$A=maple("StringTools[Randomize]():combinat[randperm]([0,1,2,3,4,5])");

```
$indexA1=switch(0,$A);

$indexA2=switch(1,$A);

$indexA3=switch(2,$A);

$indexA4=switch(3,$A);

$indexA5=switch(4,$A);

$indexA6=switch(5,$A);



$a1="Appel plukken";
$a2="Appel wassen";
$a3="Appel schillen";
$a4="Appel in stukjes snijden";
$a5="Appel opeten";
$a6="Mond afvegen";
$naam1=switch($indexA1,"$a1","$a2","$a3","$a4","$a5","$a6");
$naam2=switch($indexA2,"$a1","$a2","$a3","$a4","$a5","$a6");
$naam3=switch($indexA3,"$a1","$a2","$a3","$a4","$a5","$a6");
$naam4=switch($indexA4,"$a1","$a2","$a3","$a4","$a5","$a6");
$naam5=switch($indexA5,"$a1","$a2","$a3","$a4","$a5","$a6");
$naam6=switch($indexA6,"$a1","$a2","$a3","$a4","$a5","$a6");
$AA=maple("$A+[1,1,1,1,1,1]");
$lijst=maple("with(ListTools):[Search(1,$AA),Search(2,$AA),Search(3,$AA),Search(4,$AA),Search(5,$AA),
Search(6,$AA)]");
$L=maple("ListTools[Flatten]([$lijst,[p]])");
```

Het eerste stuk dient ervoor om de acties door elkaar te gooien.
Vervolgens hogen we de getallen in de lijst $A met eentje op om de nummers aan te duiden.
In deze lijst $AA zoeken we op welke plaats nummer 1 voorkomt, en vervolgens op welke plaats nummer 2 voorkomt enz.
De lijst $lijst geeft dus de goede volgorde aan van de acties die aangeduid worden met $naam1, $naam2 enz.

Deze lijst $lijst breiden we uit met nog een letter bijvoorbeeld er achter en deze lijst $L gebruiken we bij de *Grading Code* van de Maple-vraag.
Als u nu het invulveld gaat geprogrammeren, zorg er dan voor dat in de rubriek "Answer" bij het Maple-graded invulveld wordt ingevuld: op($lijst).
Dit is dus het correcte antwoord genoteerd als sequentie gescheiden door komma's en zonder haken.

De *Grading Code* is nu als volgt als u wilt honoreren dat een opeenvolging van 6 of 5 of 4 of 3 dedeeltelijk goedgerekend mag worden. (2 en 1 honoreren we niet.)

```
with(ListTools): R:=Flatten([[$RESPONSE],[q]]):
a:=R-$L:
b:=Rotate(R,1)-$L:
c:=Rotate(R,2)-$L:
d:=Rotate(R,3)-$L:
e:=Rotate(R,4)-$L:
f:=Rotate(R,5)-$L:
g:=Rotate(R,6)-$L:
if verify([0,0,0,0,0,0],R-$L,sublist) then m:=6
elif verify([0,0,0,0,0],b,sublist) or verify([0,0,0,0,0],g,sublist) then m:=5
elif verify([0,0,0,0],a,sublist) or verify([0,0,0,0],b,sublist) or verify([0,0,0,0],c,sublist) or verify([0,0,0,0],d,sublist) or
verify([0,0,0,0],e,sublist) or verify([0,0,0,0],f,sublist) or verify([0,0,0,0],g,sublist) then m:=4
```

```
elif verify([0,0,0],a,sublist) or verify([0,0,0],b,sublist) or verify([0,0,0],c,sublist) or verify([0,0,0],d,sublist) or verify([0,0,0],e,sublist)
or verify([0,0,0],f,sublist) or verify([0,0,0],g,sublist) then m:=3 else m:=0 end if:
grade:=evalf(m/6):
grade;
```

Het betekent dat de respons van de student omgezet wordt in een lijst, aangevuld met een letter.
Vervolgens trekken we de correcte lijst af van de lijst die de student geeft en als het goed is komen er 6 nullen te staan. De grading is dan 100% =1.
Het kan ook zijn dat de student niet de goede volgorde heeft maar wel een sub-reeks goed heeft. We roteren de lijst van de student dan net zo lang totdat die op zijn mooist samenvalt met de correcte lijst.
Als de correcte lijst bijvoorbeeld was: [2,3,6,5,1,4] en de student heeft 3,6,5,4,1,2 ingevuld, dan is het rijtje 3,6,5 alvast een goede volgorde.
Je gaat dan net zo lang roteren tot het lijstje [2,3,6,5,1,4, p] zo goed mogelijk samenvalt met een van de volgende lijstjes:

[3,6,5,4,1,2,q] (=R)
[6,5,4,1,2,q,3] (=Rotate(R,1))
[5,4,1,2,q,3,6] (=Rotate(R,2))
[4,1,2,q,3,6,5] (=Rotate(R,3))
[1,2,q,3,6,5,4] (=Rotate(R,4))
[2,q,3,6,5,4,1] (=Rotate(R,5))
[q,3,6,5,4,1,2] (=Rotate(R,6))
[2,3,6,5,1,4, p] = correcte lijst

Deze laatste twee van elkaar afgetrokken levert 3 nullen.
De grading is dan steeds het maximum van het aantal nullen gedeeld door 6.
De grading is dus 3/6 = 0.5 = 50%
Het zal nu ook duidelijk zijn waarom een extra letter aan het eind van de lijst nodig is want anders tellen de cyclische verwisselingen ook mee.

TIP: Als het een groter of kleiner lijstje is, moet u zowel het *Algorithm* aangepassen als ook de grading code!

Voor drie elementen op volgorde:

```
$A=maple("StringTools[Randomize]():combinat[randperm]([0,1,2])");

$indexA1=switch(0,$A);

$indexA2=switch(1,$A);

$indexA3=switch(2,$A);
$a1="Appel plukken";
$a2="Appel wassen";
$a3="Appel schillen";
$naam1=switch($indexA1,"$a1","$a2","$a3");
$naam2=switch($indexA2,"$a1","$a2","$a3");
$naam3=switch($indexA3,"$a1","$a2","$a3");
$AA=maple("$A+[1,1,1]");
$lijst=maple("with(ListTools):[Search(1,$AA),Search(2,$AA),Search(3,$AA)]");
$L=maple("ListTools[Flatten]([$lijst,[p]])");
```

Grading Code:

```
with(ListTools): R:=Flatten([[$RESPONSE],[q]]):
a:=R-$L:
b:=Rotate(R,1)-$L:
c:=Rotate(R,2)-$L:
d:=Rotate(R,3)-$L:
```

```
if verify([0,0,0],R-$L,sublist) then m:=3
elif verify([0,0],b,sublist) or verify([0,0],d,sublist) then m:=2
else m:=0 end if:
grade:=evalf(m/3):
grade;
```

TIP: De grading code voor 7 elementen op volgorde waarbij minder dan 4 opeenvolgingen niet meer gehonoreerd wordt:

```
with(ListTools): R:=Flatten([[$RESPONSE],[q]]):

a:=R-$L:

b:=Rotate(R,1)-$L:

c:=Rotate(R,2)-$L:

d:=Rotate(R,3)-$L:

e:=Rotate(R,4)-$L:

f:=Rotate(R,5)-$L:

g:=Rotate(R,6)-$L:

h:=Rotate(R,7)-$L:

if verify([0,0,0,0,0,0,0],R-$L,sublist) then m:=7

elif verify([0,0,0,0,0,0],b,sublist) or verify([0,0,0,0,0,0],h,sublist) then m:=6

elif verify([0,0,0,0,0],a,sublist) or verify([0,0,0,0,0],b,sublist) or verify([0,0,0,0,0],c,sublist) or verify([0,0,0,0,0],d,sublist) or
verify([0,0,0,0,0],e,sublist) or verify([0,0,0,0,0],f,sublist) or verify([0,0,0,0,0],g,sublist) or verify([0,0,0,0,0],h,sublist) then m:=5

elif verify([0,0,0,0],a,sublist) or verify([0,0,0,0],b,sublist) or verify([0,0,0,0],c,sublist) or verify([0,0,0,0],d,sublist) or
verify([0,0,0,0],e,sublist) or verify([0,0,0.0],f,sublist) or verify([0,0,0,0],g,sublist) or verify([0,0,0,0],h,sublist) then m:=4 else m:=0
end if:

grade:=evalf(m/7):

grade;
```

1.5.1 Lijstje met ListTool commando's

- **verify**(lijst1,lijst2,sublist) controleert of lijst1 een sublijst is van lijst2. Dit commando zit algemeen in Maple en niet binnen een pakket.

```
with(ListTools):
```

- **Flatten**([lijst1,lijst2]) maakt van een lijst met meer lijsten (of genestte lijsten) één lijst; handig als u een lijst wilt uitbreiden.
- **Interleave**(lijst1,lijst2) zet de elementen van de lijsten (die even lang moeten zijn) om en om achter elkaar en combineert zo tot één lijst.
- **MakeUnic(lijst)** zet de lijst om in een lijst zonder dubbelingen.
- **Reverse(lijst)** zet de elementen in de lijst in omgekeerde volgorde.
- **Rotate(lijst,3)** alle elementen van de lijst verwisselen cyclisch van plaats door 3 *achteruit* te gaan. (LET OP bij hetzelfde commando binnen StringTools is het juist *vooruit*.

- **Search(1,lijst)** zoekt de eerste de beste 1 in de lijst en geeft dan de plaats in die lijst aan.
- **SearchAll(1,lijst)** zoekt alle 1 in de lijst en geeft in een rijtje weer de plaats waar deze gevonden zijn.

In het pakket Statistics zit ook nog een commando dat goed van toepassing kan komen bij gebruik van lijsten.

- **Statistics[Count](lijst)** telt de elementen van de lijst.

Voor het randomiseren zijn de volgende commando's handig:

- **StringTools[Randomize]()**:combinat[randperm]([0,1,2,3,4,5]) geeft een permutatie van de opgegeven lijst.
- **StringTools[Randomize]()**:**combinat[randperm](10)** geeft een permutatie van de lijst [1, t/m 10].
- **Statistics[Shuffle](lijst)** geeft een random permutatie van de lijst.

```
$test1=maple("StringTools[Randomize]():combinat[randperm]([0,1,2,3,4,5])");
$test2=maple("StringTools[Randomize]():combinat[randperm](10)");
```

Variable	Value
test1	[5, 1, 2, 3, 0, 4]
test2	[4, 5, 3, 6, 10, 1, 9, 2, 7, 8]

Figure 1.56: Randomiseren van lijsten

```
> ListTools[Rotate]([5,6,7,8,9,1],2);
```

$$[7, 8, 9, 1, 5, 6]$$

```
> StringTools[Rotate]("567891",2);
```

"915678"

```
> ListTools[SearchAll](1,[3,2,1,5,4,1]);
```

$$3, 6$$

```
> ListTools[Search](1,[3,2,1,5,4,1]);
```

$$3$$

```
> Statistics[Count]([3,2,1,5,4,1]);
```

$$6$$

```
> ListTools[Flatten]([3,4,5,[6,7,8]]);
```

$$[3, 4, 5, 6, 7, 8]$$

```
> ListTools[Flatten]([[3,4,5],[6,7,8]]);
```

$$[3, 4, 5, 6, 7, 8]$$

```
> ?Statistics
> ListTools[Interleave]([a,b,c],[A,B,C],[4,5,6]);
```

$$[a, A, 4, b, B, 5, c, C, 6]$$

```
> verify([0,0],[2,3,0,0,9],sublist);
```

$$true$$

```
> ListTools[MakeUnique]([1,2,3,2,4,5,4,5,6]);
```

$$[1, 2, 3, 4, 5, 6]$$

```
> Statistics[Shuffle]([1,3,4,5]);
```

[3, 5, 4, 1]

1.6 Minder Maple

$set=maple("map(expr->convert(expr,string),[12*a,11*a*b])");

$l=switch(0,$set);

$r=switch(1,$set);

1.7 Overzicht permutaties

Als er in de rubriek *Algorithm* te veel gebruikgemaakt moet worden van voorwaarden met condition: .... dat variabelen niet gelijk aan elkaar mogen zijn of niet gelijk aan 0 of iets dergelijks, dan kan het wel eens zijn dat het systeem de juiste setting gewoon niet kan vinden.
Er gebeurt namelijk het volgende als er randomvariabelen gebruikt worden met condities:
Het systeem stelt een aantal variabelen vast, vervolgens wordt er gecontroleerd of aan de condities voldaan wordt (condition....) en zo niet, dan wordt er een nieuwe set vastgesteld, net zo lang totdat wél aan de condities voldaan is.
Als er maar beperkte mogelijkheden zijn, kan het voorkomen dat dat gewoon helemaal niet lukt en is het beter om de condities te omzeilen.

Zo is het bijvoorbeeld beter om niet het volgende te doen:

$a=range(-10,10);
condition:not(eq($a,0));

Maar beter het volgende:

$b=switch(rint(2),range(-10,-1),range(1,10));

In het algemeen zullen de getallen gebruikt gaan worden voor de indices ten behoeve van de functie switch.
Zie ook paragraaf *Overzicht Randomvariabelen (page 52).*

TIP: Maar zo min mogelijk gebruik van condition:.....

TIP: In de volgende paragrafen staan stukjes code. Het is handig om zo'n stukje code eens over te nemen in een van uw vragen in het *Algorithm* om het uit te proberen.

1.7.1 Lijstje met verschillende indices vaste volgorde

In tekstuele randomisering komt het vaak voor dat er indices gemaakt moeten worden die allemaal verschillend zijn ten behoeve van de functie switch.
Om vier *verschillende* variabelen te krijgen waarbij gekozen mag worden uit een beperkt aantal elementen kan het volgende bedacht worden:

$test=maple("combinat[permute]([0,1,2,3,4,5,6,7,8,9],4)");

Hiermee zouden we ALLE combinaties te zien krijgen van 4 elementen uit een lijst van 10.
Dat zijn er $10 \times 9 \times 8 \times 7 = 5040$ combinaties. Maar die willen we niet allemaal zien, dus kiezen we daaruit at random een *combinatie* (rijtje van 4 elementen) met het Maple-commando randcomb.

$A=maple("randomize():combinat[randcomb]([0,1,2,3,4,5,6,7,8,9],4)");

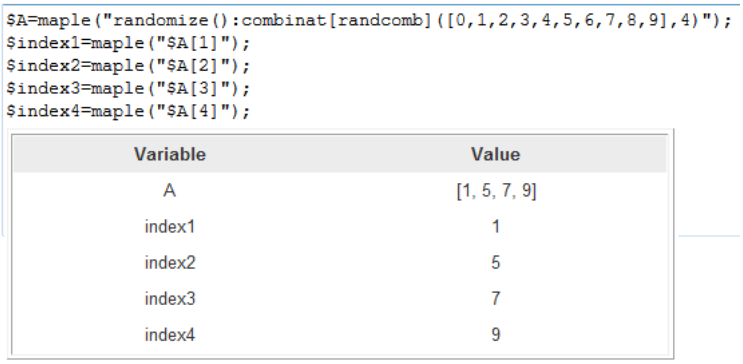
$index1=switch(0,$A);

$index2=switch(1,$A);

```
$index3=switch(2,$A);

$index4=switch(3,$A);

$AA=maple("randomize():
combinat[randcomb]([0,1,2,3,4,5,6,7,8,9],4)");

$A=maple("op($AA)");

$indexA1=switch(0,$A);

$indexA2=switch(1,$A);

$indexA3=switch(2,$A);

$indexA4=switch(3,$A);
```

Op deze manier krijgt u dus vier *verschillende* variabelen uit het lijstje van [0,1,2,3,4,5,6,7,8,9].

TIP: Helaas is het lijstje dat dan geleverd wordt ALTIJD gerangschikt in dezelfde volgorde waarmee ook de oorspronkelijke lijst wordt aangeboden. Maar soms is dat geen bezwaar als u toch gebruikmaakt van de optie *Permute* van een *Multiple Choice*-vraag bijvoorbeeld.



```
$A=maple("randomize():combinat[randcomb]([0,1,2,3,4,5,6,7,8,9],4)");
$index1=maple("$A[1]");
$index2=maple("$A[2]");
$index3=maple("$A[3]");
$index4=maple("$A[4]");
```

Variable	Value
A	[1, 5, 7, 9]
index1	1
index2	5
index3	7
index4	9

Figure 1.57: Combinatie van 4 elementen kiezen uit 10 op volgorde van klein naar groot

Met de functie switch kunnen de nodige variabelen aangemaakt worden als de indexering nu vast ligt.

TIP: Let op dat vooraf randomize(): gegeven moet worden, anders wordt er steeds hetzelfde gekozen. Het heeft met de inwendige klok te maken voor de setting van de randomisering.

TIP: Als er twee *verschillende* combinaties at random gegenereerd moeten worden uit *dezelfde* beginlijst, is het handig om een randomgetal bij randomize() mee te geven. Als u dat niet doet, dan krijgt u twee identieke lijstjes omdat de randomisering van het *Algorithm* met Maple op hetzelfde tijdstip start.

```
$getalA=range(1,1000);
$getalB=range(1,1000);
condition:not(eq($getalA,$getalB));
$A=maple("randomize($getalA):combinat[randcomb]([0,1,2,3,4,5,6,7,8,9,10,11],4)");

$indexA1=switch(0,$A);

$indexA2=switch(1,$A);

$indexA3=switch(2,$A);

$indexA4=switch(3,$A);
$B=maple("randomize($getalB):combinat[randcomb]([0,1,2,3,4,5,6,7,8,9,10,11],4)");
```

```
$indexB1=switch(0,$B);
$indexB2=switch(1,$B);
$indexB3=switch(2,$B);
$indexB4=switch(3,$B);
```

TIP: Nog steeds zullen deze gegenereerde lijstjes in de volgorde van de aangeboden lijst staan.
In de volgende paragraaf ziet u hoe u de volgorde eventueel nog door elkaar kunt husselen.

1.7.2 Lijstje met verschillende indices willekeurige volgorde

TIP: $test=maple("combinat[randperm](10)");
Hiermee maakt u een lijstje van de getallen 1 t/m 10 in willekeurige volgorde.

Het is mogelijk om met permutaties te werken om een aantal verschillende elementen in een lijstje op *willekeurige volgorden* te krijgen.

```
$lijstA=maple("[0,1,2]");
$A=maple("StringTools[Randomize]():combinat[randperm]($lijstA)");
$indexA1=switch(0,$A);
$indexA2=switch(1,$A);
$indexA3=switch(2,$A);
$Antwg1=switch($indexA1,"het bestemmingsplan", "de bouwverordening","het Bouwbesluit");
$Antwg2=switch($indexA2,"het bestemmingsplan", "de bouwverordening","het Bouwbesluit");
$Antwg3=switch($indexA3,"het bestemmingsplan", "de bouwverordening","het Bouwbesluit");
```

Met de eerste regel wordt een lijstje aangemaakt.
In de tweede regel wordt de oorspronkelijke lijst at random gepermuteerd met behulp van het Maple-commando randperm.
Echter wilt u niet steeds dezelfde permutatie hebben, dan moet u vooraf vreemd genoeg StringTools[Randomize]() meegeven. (Het is niet echt zoals het hoort dat de StringTools erbij gehaald moeten worden maar het is een fout in de programmering van Maple waar bepaalde routines geleend zijn van het StringTools-pakket. Het werkt in ieder geval wel. Het is overlegd met MapleSoft en er wordt aan gewerkt om het zonder StringTools ook goed te laten lopen in de toekomst.)
Vervolgens kunt u indices maken die gebruikt kunnen worden bij de functie switch en u weet zeker dat u in dit geval drie dingen krijgt die een willekeurige volgorde hebben en allemaal verschillend zijn: $Antwg1, $Antwg2 en $Antwg3.
Zie ook *Figure 1.17 (page 11)*.

```
$lijstA=maple("[0,1,2]");
$A=maple("StringTools[Randomize]():combinat[randperm]($lijstA)");
$indexA1=maple("$A[1]");
$indexA2=maple("$A[2]");
$indexA3=maple("$A[3]");
$Antwg1=switch($indexA1,"het bestemmingsplan", "de bouwverordening","het
Bouwbesluit");
$Antwg2=switch($indexA2,"het bestemmingsplan", "de bouwverordening","het
Bouwbesluit");
$Antwg3=switch($indexA3,"het bestemmingsplan", "de bouwverordening","het
Bouwbesluit");
```

Variable	Value
lijstA	[0, 1, 2]
A	[1, 2, 0]
indexA1	1
indexA2	2
indexA3	0
Antwg1	de bouwverordening
Antwg2	het Bouwbesluit
Antwg3	het bestemmingsplan

Figure 1.58: Permutatie van een lijst

1.7.3 Aantal elementen kiezen uit een grotere lijst met willekeurige volgorde

Op de volgende manier genereert u een lijst van bijvoorbeeld 4 elementen in willekeurige volgorde gekozen uit een grotere lijst van bijvoorbeeld 5 elementen.

```
$AA=maple("randomize():combinat[randcomb]([0,1,2,3,4],4)");
$A=maple("StringTools[Randomize]():combinat[randperm]($AA)");

$index1=switch(0,$A);

$index2=switch(1,$A);

$index3=switch(2,$A);

$index4=switch(3,$A);
```

In de eerste regel wordt er dus een *combinatie* gegenereerd van 4 elementen, gekozen uit een lijst van 5 elementen. Het resultaat ervan is een combinatie waarbij de volgorde altijd van klein naar groot geordend is.
Vergeet niet bij deze opdracht randomize() mee te geven omdat anders altijd dezelfde lijst zou ontstaan. Het heeft te maken met de setting van de inwendige klok van het systeem, waar de randomisering begint.
Vervolgens wordt deze lijst doorelkaar gehusseld en wordt er dus at random een permutatie gegenereerd met het Maple-commando combinat[randperm](...).

```
$AA=maple("randomize():combinat[randcomb]([0,1,2,3,4],4)");
$A=maple("StringTools[Randomize]():combinat[randperm]($AA)");
$index1=maple("$A[1]");
$index2=maple("$A[2]");
$index3=maple("$A[3]");
$index4=maple("$A[4]");
```

Variable	Value
AA	[0, 1, 3, 4]
A	[4, 1, 0, 3]
index1	4
index2	1
index3	0
index4	3

Figure 1.59: Permuteren van een lijst gekozen uit een grotere lijst

TIP: Als het in feite alleen maar gaat om het genereren van indices, kunt u ook volstaan met het volgende script:

```
$AA=maple("[0,1,2,3,4]");
$A=maple("StringTools[Randomize]():combinat[randperm]($AA)");
$index1=maple("$A[1]");
$index2=maple("$A[2]");
$index3=maple("$A[3]");
$index4=maple("$A[4]");
```

Variable	Value
AA	[0, 1, 2, 3, 4]
A	[3, 2, 4, 0, 1]
index1	3
index2	2
index3	4
index4	0

Figure 1.60: Indices maken gekozen uit een grote lijst

TIP: Als u minder gebruik wilt maken van het aanroepen van maple, dan kunt u het volgende doen:

$AA=maple("StringTools[Randomize]():combinat[randperm]([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19])");

$A=maple("op($AA)");

$indexA1=switch(0,$A);

$indexA2=switch(1,$A);

Nu volgt nog een voorbeeld van het genereren van 4 even getallen en twee oneven getallen in willekeurige volgorde.

```
$even=maple("[2,4,6,8,10,12,14,16,18,20,22,24]");
$Aeven=maple("StringTools[Randomize]():combinat[randperm]($even)");
$oneven=maple("[1,3,5,7,9,11,13,15,17,19,21]");
$Boneven=maple("StringTools[Randomize]():combinat[randperm]($oneven)");
$a=maple("$Aeven[1]");
$b=maple("$Aeven[2]");
$c=maple("$Aeven[3]");
$d=maple("$Aeven[4]");
$e=maple("$Boneven[1]");
$f=maple("$Boneven[2]");
```

Met hieronder het effect ervan:

$even=maple("[2,4,6,8,10,12,14,16,18,20,22,24]");
$Aeven=maple("StringTools[Randomize]():combinat[randperm]($even)");
$oneven=maple("[1,3,5,7,9,11,13,15,17,19,21]");
$Boneven=maple("StringTools[Randomize]():combinat[randperm]($oneven)");
$a=maple("$Aeven[1]");
$b=maple("$Aeven[2]");
$c=maple("$Aeven[3]");
$d=maple("$Aeven[4]");
$e=maple("$Boneven[1]");
$f=maple("$Boneven[2]");

Variable	Value
even	[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24]
Aeven	[16, 6, 8, 14, 18, 12, 24, 20, 4, 10, 2, 22]
oneven	[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21]
Boneven	[3, 13, 19, 11, 17, 5, 1, 21, 7, 9, 15]
a	16
b	6
c	8
d	14
e	3
f	13

Figure 1.61: Even en oneven getallen genereren

TIP: Als u slechts twee indices nodig heeft is het erg eenvoudig om het volgende te doen:
$index1=rint(2);
$index2=not($index1);

TIP: In feite kunt u dus volstaan om alleen het volgende te gebruiken in de meeste gevallen:

$lijstA=maple("[.....]");
$A=maple("StringTools[Randomize]():combinat[randperm]($lijstA)");
$indexA1=switch(0,$A);
$indexA2=switch(1,$A);

$indexA3=switch(2,$A);

1.8 Overzicht Randomvariabelen

1.8.1 Gebruik van de Designer in het Algorithm

Er is in deze opgave een aantal variabelen gedeclareerd in de rubriek *Algorithm*.
Allereerst de twee randomgetallen *a* en *b* aangegeven met $a en $b.
Als u de code nog niet goed kent om een variabele te declareren, kunt u dit het beste doen met behulp van de zogenoemde *Designer* van het *Algorithm* waarbij u kunt instellen om wat voor soort randomgetal het moet gaan. De code wordt dan vanzelf gegenereerd.

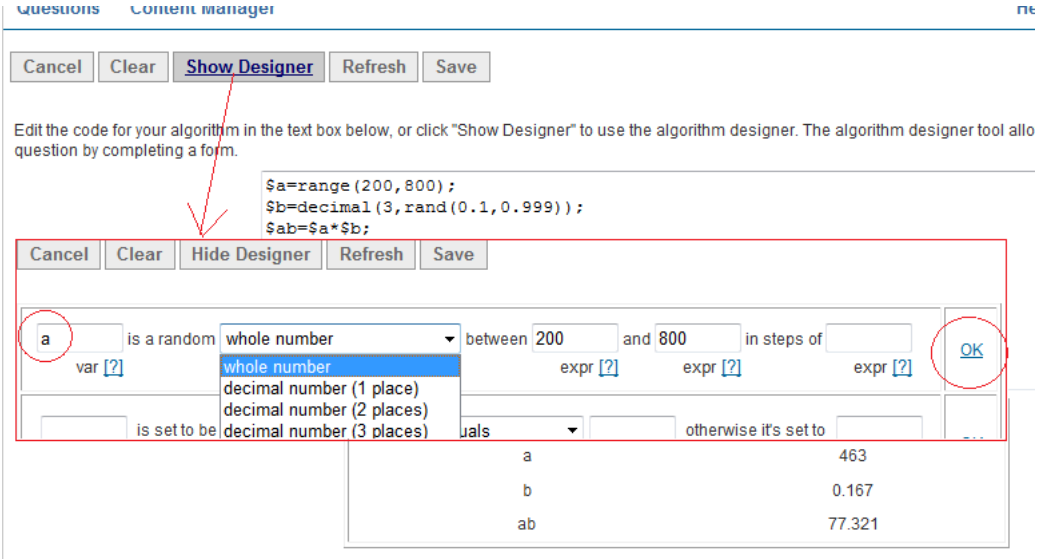


Figure 1.62: De Designer van het Algorithm

De variabele hier $ab genoemd, is gewoon de berekening $a*$b die uitbesteed wordt aan het systeem dat gewone berekeningen feilloos aan kan.

TIP: Als u in de rubriek *Algorithm* zit, klik dan op *Show Designer* om deze open te klappen. Na gebruik kunt u weer op *Hide Designer* klikken om ruimte in het scherm te krijgen.

TIP: U geeft aan de variabele in de *Designer* een naam (zonder $-teken).

Het aantal decimalen kan ingesteld worden en ook eventueel de stapgrootte. Als u bij de stapgrootte niets invult, dan is de stapgrootte bij gehele getallen gewoon 1 en als u gekozen hebt voor decimal number (3 places) bijvoorbeeld, dan is de stapgrootte gelijk aan 0.001. Verder kunt u elke gewenste stapgrootte instellen.
De code wordt na het klikken op *OK* automatisch gegenereerd.

In de volgende regel van de *Designer* is er nog de mogelijkheid om een en ander te programmeren op een makkelijke manier.

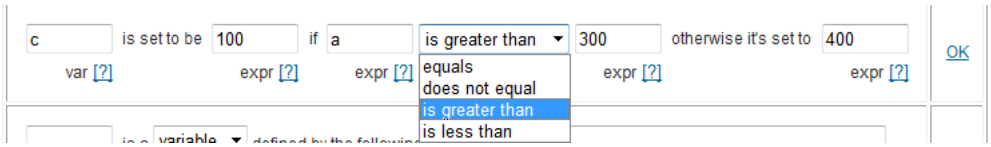


Figure 1.63: Met de Designer codes programmeren voor Algorithm variabelen

Hier wordt een variabele $c gemaakt die gelijk is aan 100 als $a groter dan 300 is en anders is $c dus gelijk aan 400.
Dit resulteert in de volgende code:

$c=if(gt($a,300),100,400);

TIP: Met de knop *Refresh* kunnen de in de rubriek *Algorithm* geprogrammeerde variabelen weer nieuwe waarden aannemen om te kijken of alles naar wens is.

TIP: Let op de volgorde van de programmeerregels.
Deze worden namelijk achter elkaar doorlopen in de volgorde waarin ze staan.

TIP: In de *Handleiding Items Maken deel B* vindt u in paragraaf *Formules met Möbius* nog meer informatie om berekeningen te doen met formules.

Extra condities kunnen nog aan een variabele toegekend worden.
Deze kunnen gemakkelijk geprogrammeerd worden vanuit de *Designer* (zie laatste rubriek van de *Designer*).
Als bijvoorbeeld de randomvariabele $ab niet gelijk mag zijn aan 1.

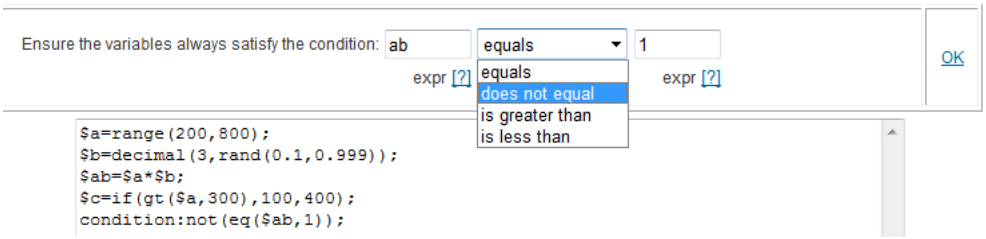


Figure 1.64: Condities van variabelen met de Designer

TIP: Merk op dat het in de *Designer* niet nodig is dat het dollarteken erbij getikt wordt als het over de variabele $ab gaat bijvoorbeeld. Dat dollarteken komt er in de code automatisch wel bij te staan.

BELANGRIJK!! Gebruik zo weinig mogelijk van deze condities. Wat er in feite gebeurt is dat uit de range een waarde gekozen wordt. Als deze waarde niet voldoet aan de conditie, dan wordt er een nieuwe waarde gekozen, net zolang totdat wel aan de voorwaarde voldaan is.
Worden er teveel condities meegegeven, dan kan het zijn dat het systeem te vaak opnieuw de waarden voor de variabelen moet genereren en dan kan dat uitmonden in een system overload. Er zijn vele andere mogelijkheden om een set variabelen direct te genereren met de nodige condities. Kijk ook eens in paragraaf *Overzicht Permutaties* *(page 47)* bij de permutaties en combinaties of gebruik de volgende TIP.

TIP: Wat ook mogelijk is om uit meerdere ranges te kiezen met switch zoals bijvoorbeeld:
$c=switch(rint(2),range(-9,-1),range(1,9));
Het betekent dat er om de beurt tussen de twee ranges geswitcht wordt.
Zie ook bij switch en rint verderop in paragraaf *Randomgetallen* *(page 54)*.

TIP: Voor de zekerheid zet u altijd haakjes om de variabelen in de code als deze variabelen soms ook negatief kunnen zijn.

TIP: vergeet in de *Designer* niet om rechts op *OK* te klikken voor bevestiging!
Meestal is de rechterkant van het scherm namelijk niet zichtbaar als het om smalle schermen gaat!

Met de *Designer* kunt u ook een plotcommando invoeren en dan wordt dat vertaald in het *Algorithm*.

Figure 1.65: Een grafiek voorbereiden in het Algorithm

In de *Designer* is ingevuld dat het een plot moet zijn en bij het Maple-commando is ingevuld: plot($f, x=-10..10,-20..20,thickness=2,discont=true,color=$kleur)
De codering die dan automatisch gemaakt wordt is:
$grafiek=plotmaple("plot($f, x=-10..10,-20..20,thickness=2,discont=true,color=$kleur) ");
Het is handig als u zelf nog achteraf in de code de afmetingen van de grafiek erbij geeft, want default is de grafiek aan de grote kant.

De code wordt dan bijvoorbeeld:

```
$grafiekf=plotmaple("plot($f, x=-10..10,-20..20,thickness=2,discont=true,color=$kleur),
 plotoptions='height=200, width=200' ");
```

Zie ook paragraaf *Matching met formules en dynamische grafieken* *(page 34)*.
Voor meer informatie over dynamische figuren, zie in de Handleiding Möbius deel B in paragraaf *Dynamische figuren*.

TIP: Als u gebruik wilt maken van *Repositories*, is er nog een aparte handleiding voor dat onderwerp. U moet daarvoor wel het computeralgebrasysteem Maple op uw eigen machine geïnstalleerd hebben.

1.8.2 Randomgetallen

Mogelijke randomgetallen zijn te maken op de volgende manieren:

- range(-10,10)
  geeft een geheel getal tussen -10 en 10 inclusief eindpunten (met stappen van 1).
- range(3.5,10)
  geeft de getallen 3.5 of 4.5 tot en met 9.5 (dus altijd stappen van 1).
- range(-10,10,3)
  geeft een geheel getal tussen -10 tot en met 10 met stappen van 3.
  Dus -10, -7, -4 enz. tot en met 8.
- switch(rint(2),range(-10,-1),range(1,10)); om te voorkomen dat 0 ook meegenomen wordt. Dat spaart een conditie uit!
- range(3.5,15,3)
  geeft getallen vanaf 3.5 , 6.5 enz. in stappen van 3.
- range(1000,5000)
  geeft een getal tussen 1000 en 5000 bijvoorbeeld 1997 maar dit getal wordt als 1,997 gepresenteerd!
  In berekeningen gaat het verder wel goed met de separator voor duizendtallen.
  TIP: Als dit getal echter wordt gebruikt op het scherm in de presentatie van de vraag, kan het verwarring veroorzaken een aangezien worden voor een decimale komma in plaats van een decimale punt.
  Maak in dat geval een presentabel getal zonder duizendtalseparator, eventueel met behulp van Maple:
  $a=range(1000,5000);
  $A=maple("$a"); of $A=mathml("$a"); of $A="$a";
  met $A kunt u dan het getal in de opgave en in de *Feedback* presenteren en komt het er zonder komma (duizendtalseparator) te staan.
- rand(30, 50.67)
  geeft een willekeurig reëel getal tussen 30 en 50.67 (vaak met 6 cijfers achter de decimale punt) met eindpunt meegerekend.
- rand(30, 50.67,4)
  geeft een reëel getal tussen 30 en 50.67 met 4 *significante* cijfers met eindpunt meegerekend.
- decimal(2,rand(1,10))
  geeft een decimaal getal met twee cijfers achter de decimale punt tussen 1 en 10.
- decimal(2,range(3.5,20.8,0.03))
  geeft een decimaal getal met twee cijfers achter de decimale punt tussen 3.5 en 20.8 met stappen van 0.03
  TIP: Met range is de stapgrootte in te stellen en met rand is het aantal significante cijfers in te stellen.
  decimal(2, range(1,10)) kan dus niet, want bij range is de stapgrootte altijd 1 en dan krijgt u nooit een decimaal getal.
- numfmt("#.00",20.9) geeft als resultaat 20.90 (numeriek volgens format).
  U kunt ook Maple inschakelen om de afronding te doen met maple("Float(round(100*20.9),-2);"); dat 20.90 oplevert.
  numfmt(#0.000,3/4) geeft als resultaat 0.750.
  U kunt ook Maple inschakelen om de afronding te doen met maple("Float(round(1000*3/4),-3);"); dat 0.750 oplevert.
- sig(3, 20.8571)
  geeft in dit geval 20.9. Het is een afronding naar 3 significante cijfers.

- int(20.8571)
  geeft in dit geval 20. Het kapt af op gehele getallen.
  U zou ook decimal(0,20.8571) kunnen doen, maar dan wordt het 21, een afronding dus.
  condition: not(eq($c/$d,int($c/$d))); betekent dat de deling $d/$d niet een geheel getal oplevert.
- switch(3,a,b,c,d)
  geeft de letter d.
  Het getal 3 fungeert hier als index en de index van switch loopt namelijk van 0 tot aan het aantal elementen.
  Zijn er dus *n* elementen, dan loopt de index van 0 tot en met *n* - 1.
  Mooie voorbeelden met switch zijn te vinden in paragraaf Randomiseren met switch *(page 2)*.
- switch(rand(0,5), 2, 3, 5,6,7,8)
  Geeft één uit de opgenoemde getallen.
  rand(0,5) heeft hier de functie van index en loopt van 0 tot en met 5 (dus 6 mogelijkheden).
- switch(rint(6),2,3,5,6,7,8)
  geeft hetzelfde effect als hierboven: switch(rand(0,5), 2, 3, 5,6,7,8).
- rint(6)
  genereert een geheel getal uit de rij 0, 1, 2, 3, 4, 5
  Dus rint(n) betekent een random integer 0 tot en met *n* - 1.
- switch(rint(2),range(-5,-1),range(1,5))
  Hiermee kunt u snel een variabele uit het interval [-5,5] kiezen, waarbij 0 uitgesloten wordt.
- lsu(3, 48.9876) (de eerste letter is de l van letter).
  geeft 0.1: de kleinste significante eenheid van het getal op de derde plaats (least significant unit).
  lsu(2,0.00589) geeft 0.0001
  lsu(3,12765.987) geeft 100.
  Dit is erg gemakkelijk bij het bepalen van de tolerantie van een numeriek antwoord bij het vraagtype *Formula*. Daar kan namelijk de tolerantie aangegeven worden met een vraagteken en van te voren kan de variabele voor de tolerantie in het *Algorithm* aangemaakt worden. MapleSoft maakt het ook mogelijk om een variabele tolerantie aan te geven bij het numerieke vraagtype.
- $a=if(ne(($a1),($b)),($a1),($a1)+1)
  Betekent: als $a1 niet gelijk is aan $b, dan wordt de variabele $a gelijk aan $a1 en anders $a1+1.
  Deze regel kan gebruikt worden als alternatief voor bijvoorbeeld de volgende conditie.
- condition:ne($a,$b)
  Betekent dat de variabelen $a en $b niet gelijk mogen zijn. Deze variabele kan gemaakt worden met de *Designer*.
  TIP: Gebruik niet teveel "condition" in uw *Algorithm*!
- condition:gt($teller,$noemer)
  Betekent: voorwaarde dat de $teller groter is dan de $noemer.
  Eventueel gt vervangen door eq (gelijk) of lt (kleiner dan) of not(eq($teller,$noemer)).
  Deze variabele kan eventueel ook gemaakt worden met de *Designer*.
- $b=if(lt($a,5),20,30); betekent, dat als $a kleiner dan of gelijk is aan 5, dat $b is 20 en anders is $b gelijk aan 30.
  Deze variabele kan eventueel ook gemaakt worden met behulp van de *Designer*.
- $X=if(lt($x,-180), $x+360, if(gt($x,180), $x-360,$x)); betekent als $x kleiner is dan 180 dan moet er 360 bij opgeteld worden en als $x groter dan 180 dan moet er 360 van afgetrokken worden en anders blijft het $x.
- $a=maple("randomize():RandomTools[Generate](choose(remove(has,[seq(seq(i*k^2,i=2..10),k=2..10)],[seq(k^2,k=2..31)])))");
  Op deze manier is het mogelijk om zelf een range te maken met bepaalde getallen waaruit gekozen kan worden.
  Van alle getallen in de lijst van de vorm i*k^2 waarbij i = 2 t/m 10 en k =2 t/m10, wordt gecontroleerd of er kwadraten, dus elementen in zitten van de lijst getallen k^2 waarbij k loopt van 2 t/m 31, [seq(k^2,k=2..31)] en zo ja, dan worden die er uitgehaald met remove, en vervolgens wordt er eentje uitgekozen met choose.
  Bij gebruik van RandomTools moet altijd eerst randomize() gegeven worden, anders wordt steeds dezelfde waarde gegenereerd.
  remove is een toplevel-commando en Generate(choose([lijst])) is een commando uit RandomTools-pakket.
  Zie ook in paragraaf *Randomiseren met switch (page 2)* voor het willekeurig kiezen uit een of meer verzamelingen opgegeven getallen. Zie ook paragraaf *Overzicht Permutaties (page 47)*.

- min en max geven respectievelijk het minimum en het maximum van een rijtje getallen. Let op dat deze altijd vertaald worden naar numerieke waarden als deze irrationaal zijn. (Werk eventueel met Maple.)
- max(remove(has,[$g1,$g2,$g3,$g4,$g5,$g6],$grootste))
  Als eerst $grootste is gedefinieerd als grootste van betreffende lijst getallen, dan kan deze grootste eruit weggehaald worden en vervolgens kan dan weer de grootste gekozen worden.
- $a=switch(rint(4),6,7,8,0);
  $b=not($a);
  Dit betekent dat steeds $a een getal is uit het rijtje en dat $b gelijk is aan 0 óf als $a gelijk is aan 0, dan is $b gelijk aan1.
- randcomb is een Maple-commando uit het combinat-pakket en genereert een random combinatie uit een lijst of verzameling. Deze gegenereerde lijst staat altijd in volgorde van klein naar groot.
  $AA=maple("randomize():combinat[randcomb]([0,1,2,3,4],4)"); geeft bijvoorbeeld [0,2,3,4].
- randperm is een Maple-commando uit het combinat-pakket en genereert een random permutatie van een lijst. Gooit dus de volgorde doorelkaar.
  $A=maple("StringTools[Randomize]():combinat[randperm]([0,2,3,4])"); geeft bijvoorbeeld [4,0,3,2].
- randomize() en StringTools[Randomize] () zijn Maple-commando's die ervoor zorgen dat steeds een andere combinatie of permutatie wordt gegenereerd.
- Search is een Maple-commando uit het pakket ListTools en kan een element zoeken.
  maple("ListTools[Search](1,[4,5,6,1,2,3])"); geeft de plaats waar de 1 zich bevindt en zal dus leveren het getal 4.
- SearchAll is een Maple-commando uit het pakket ListTools en kan elementen zoeken.
  maple("ListTools[SearchAll](1,[0,1,0,0,1,0])"); geeft alle plaatsen waar de 1 zich bevindt en zal hier dus leveren het rijtje getallen: 2,5
- Maple in combinatie met switch. Met Maple eerst een index aanmaken en vervolgens deze index gebruiken voor de functie switch:
  $g1=maple("if $a>0 then 0 else 1 end if ");
  $g=switch($g1,"+","-");
- Als boven komt op hetzelfde neer in één regel:
  $g=if(gt($a,0),"+","-");

Index